(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 696 29 800 T2** 2004.04.29

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 813 713 B1**
(21) Deutsches Aktenzeichen: **696 29 800.7**
(86) PCT-Aktenzeichen: **PCT/US96/02384**
(96) Europäisches Aktenzeichen: **96 909 505.8**
(87) PCT-Veröffentlichungs-Nr.: **WO 96/027834**
(86) PCT-Anmeldetag: **29.02.1996**
(87) Veröffentlichungstag der PCT-Anmeldung: **12.09.1996**
(97) Erstveröffentlichung durch das EPA: **29.12.1997**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **03.09.2003**
(47) Veröffentlichungstag im Patentblatt: **29.04.2004**

(51) Int Cl.[7]: **G06F 12/10**
G06F 17/30

(30) Unionspriorität:
397809 **03.03.1995** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Fujitsu Ltd., Kawasaki, Kanagawa, JP

(74) Vertreter:
W. Seeger und Kollegen, 81369 München

(72) Erfinder:
PENG, Kuo-Liang, Leon, Mountain View, US; LIH, Yolin, San Jose, US; CHANG, David, Chi-Wei, Saratoga, US

(54) Bezeichnung: **ADRESSENÜBERSETZUNGSBUFFER IN EINEM RECHNERSYSTEM**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

HINTERGRUND DER ERFINDUNG

Gebiet der Erfindung

[0001] Diese Offenbarung betrifft Speicherverwaltungseinheiten, insbesondere Speicherverwaltungseinheiten, die einen Adreßumsetzpuffer enthalten, der verwendet wird, um eine Übersetzung in einem Computersystem zu beschleunigen, das einen virtuellen Speicher unterstützt, und insbesondere Verfahren zum Verhindern und Wiederherstellen einer Situation, in der Mehrfachübersetzungen in dem Puffer derselben Adresse entsprechen. Zusätzlich betrifft diese Offenbarung eine spezielle Datenstruktur, die verwendet wird, um Übersetzungen zu speichern, d. h. einen B-Baum, auf den zugegriffen wird, wenn eine angefragte Übersetzung nicht in dem Seitenblickpuffer vorhanden ist.

Technischer Hintergrund der Erfindung

[0002] In Computern, die ein virtuelles Speichersystem unterstützen, wird der Adressenraum, auf den sich Programme beziehen, "virtueller Speicher" genannt, und jede virtuelle Adresse, die durch einen Programmbefehl spezifiziert ist, wird von der Speicherverwaltungseinheit (MMU) in eine physikalische oder reale Adresse übersetzt, die zum Hauptspeicheruntersystem (nachfolgend bezeichnet als "Speicher") weitergegeben wird, um den Posten wieder zu erlangen, auf den zugegriffen wurde. Die Verwendung von virtuellem Speicher gestattet es, dass die Größe von Programmen die Größe des physikalischen Speichers weit übersteigt und stellt eine Flexibilität bei der Anordnung von Programmen in dem physikalischen Speicher bereit. Aus verschiedenen Gründen, einschließlich dem Erfordernis, Tabellen, die zur Adressenübersetzung erforderlich sind, auf einer angemessenen Größe zu halten, bewirken einige Virtuell-in-Realadressenübersetzungsschemata eine Übersetzung in zwei oder mehr Stufen.

[0003] Üblicherweise erfordert jede Stufe der Übersetzung einen oder mehrere Zugriffe auf eine Tabelle, die in einem Speicher gehalten ist. Um die Gesamtanzahl von Speicherzugriffen, die pro Adressenübersetzung erforderlich sind, zu verringern, sind oft ein oder mehrere Adreßumsetzpuffer (TLBs) in der MMU vorgesehen, um die Durchschnittszeit zu verringern, die erforderlich ist, um eine entsprechende Anzahl von Stufen in dem Adressenübersetzungsschema zu bewirken. Ein TLB ist ein cache-ähnlicher Speicher, der typischerweise in einem statischen Direktzugriffsspeicher (SRAM) und/oder einem Assoziativspeicher (CAM) implementiert ist, der Übersetzungen hält, die einer bestimmten Stufe des Übersetzungsschemas entsprechen, die kürzlich von dem Speicher geholt wurde.

[0004] Ein Zugriff auf einen TLB-Eintrag, der eine Ausgangsadresse hält, die einer Eingangsadresse entspricht, beseitigt das Erfordernis für einen und ist typischerweise viele Größenordnungen schneller als ein Zugriff auf die Eingangsspeichertabelle, um die Ausgangsadresse wieder zu erlangen, die der Eingangsadresse entspricht. Ein TLB-Eintrag kann Felder enthalten, die die Übersetzung beschreiben, zusätzlich zu Eingangs- und Ausgangsadressenfeldern, wie ein Schutzfeld.

[0005] Ferner kann ein oder können mehrere Felder, die verwendet werden, um die Ausgangsadresse zu bestimmen, anstelle der Ausgangsadresse selbst in den TLB-Einträgen gespeichert sein.

[0006] Wenn der TLB die angeforderte Übersetzung nicht enthält (d. h. nach einem TLB "Fehlschlag"), dann initiiert die MMU eine Durchsuchung von Übersetzungstabellen, die im Speicher gespeichert sind, nach der angefragten Übersetzung und lädt sie dann in den TLB, wo sie für einen nachfolgenden schnellen Zugriff verfügbar ist, falls eine Übersetzung derselben Eingangsadresse an einer späteren Stelle erforderlich sein sollte. Der Teil der MMU, der diese Funktion hardwaremäßig (Logikschaltung) ausführt, wird nachfolgend als der "Tabellenläufer" bezeichnet.

[0007] Aufgrund von Fehlern verschiedener Arten, wie weiche Fehler im RAM, Hartwaretransientenfehler und Softwarefehler, können zwei oder mehr Übersetzung derselben Eingangsadresse in dem TLB auftreten. Es wäre für die MMU wünschenswert, zu detektieren, dass zwei oder mehr Übersetzungen in dem TLB für die spezifizierte Eingangsadresse existieren, die übersetzt werden soll, und in der Lage zu sein, von dieser anormalen Situation zurückzufinden, indem geeignete Maßnahmen ergriffen werden, wie Invalidieren der zwei oder mehr Übersetzungen und Initiieren einer Suche durch den Tabellenläufer.

[0008] Der Eingangsadressenbereich für den Eingang zu einer speziellen Stufe des Adressenübersetzungsschemas in einem Computersystem, das einen virtuellen Speicher unterstützt, kann extrem groß sein. Zum Beispiel wird in einer 64-Bit-Arbeitsstation, die von HaL Computersystems, Inc. (Rechtsnachfolger dieser Offenbarung) verkauft wird, eine 51-Bit-Adresse in der ersten Stufe übersetzt. Eine einfache Reihe mit einem Eintrag für jede mögliche Eingangsadresse, wie sie üblicherweise beim Stand der Technik (z. B. Seitentabelle) verwendet wird, ist keine machbare Lösung hinsichtlich Speicheranforderungen zum Implementieren der Übersetzungstabelle für einen solchen großen Eingangsadressenbereich.

[0009] Bekannte Datenstrukturen zum Implementieren der Übersetzungstabelle haben Speicheranforderungen proportional zur Gesamtanzahl von möglichen Eingangsadressen statt der Anzahl von Eingangsadressen, die übersetzt wurden, und sind somit für sehr große Eingangsadressenräume nicht praktikabel.

[0010] Computing Surveys, Vol. 11, Nr. 2, Juni 1979, Seiten 121 bis 137, "The Ubigitous B-Tree", Douglas

Comer, stellt eine allgemeine Diskussion von B-Baum-Strukturen bereit. Die EP 453 707 (IBM), und Proceedings of the 1978 Conference on parallel processing, Seiten 235/236, "Concurrent accesses of B*-Trees in a paging enviroment", J. -L. Baer und S. R. Anstead, und Information Processing Letters, Vol. 34, Nr. 5, Mai 1990, NL, Seiten 251–254, "Slow Optimally Balanced Search Strategies vs. Cached Fast Uniformly Balanced Search Strategies", William Pugh, offenbaren die Verwendung von Arten von "Puffern" oder "Caches" mit seitenweisen B-Bäumen. In Comer, Baer und EP werden mehrere Knoten des B-Baums als eine Seite geladen. IBM Technical Disclosure Bulletin, 30 (1987) Nov., Nr. 6., Armonk, NY, USA, Seiten 145/146, "Software Table Look-up Buffer for B-Tree Implementation", offenbart die Verwendung eines Softwarenachschlagepuffers mit einem seitenweisen B-Baum. Die WO 94/27222 (LIEDTKE JOCHEN) offenbart einen Adressenübersetzungspuffer für eine Tabelle mit einem B-Baum.

[0011] Die US-A- 5 230 045 (SINDHU) offenbart ein System, um Duplikate in einem Adressenübersetzungspuffer zu vermeiden, in dem, wenn ein Prozessor eine VP (virtuelle Seitenadresse) und eine RP (reale Seitenadresse) und einen Satz von Markierungen bei einem Befehl bereitstellt, der für einen Eintrag anfordert, dass er in einen Adressenübersetzungspuffer ("Abbild-Cache") geschrieben werden soll, zunächst ein Versuch unternommen wird, die VP des "neuen" Eintrages mit VP-Adressen zu matchen, die bereits in dem Puffer gespeichert sind. Wenn ein Matchen gefunden wird, wird der "alte" matchende Eintrag durch den "neuen" Eintrag überschrieben. Wenn kein Matchen gefunden wird, wird ein anderer "Opfer"-Eintrag zum Überschreiben ausgewählt.

[0012] Die US-A-4 473 878 (ZOLNOWSKY) offenbart eine "globale Kollisionsüberprüfung", bei der vor dem Laden in einen Adressenübersetzungspuffer jeder "neue" Segmentdescriptor geprüft wird, um sicherzustellen, dass er nicht widerspricht oder kollidiert mit einem "alten" Segmentdescriptor, der bereits geladen ist. Die "globale Kollisionsüberprüfungs"-Prozedur ist in dem Zusammenhang des anfänglichen sequenziellen Ladens des Puffers (Segmentabbilder in MMU's) mit Segmentdescriptoren beschrieben.

[0013] Die US-A-3 800 286 (BROWN) offenbart ein System, bei welchem im Fall von Mehrfachtreffern – d. h., wenn ein Segmentdescriptor oder eine relative Adresse auftritt, so dass sie mehrmals in einem Adressenübersetzungspuffer vorhanden ist -, werden alle Informationen, die in den Stellen des Puffers enthalten sind, gelöscht. Das heißt, dass der Speicher "geräumt" ist.

ÜBERBLICK

[0014] Gemäß der vorliegenden Erfindung werden ein Verfahren gemäß Anspruch 1 und ein Gerät gemäß Anspruch 9 geschaffen.

KURZE BESCHREIBUNG DER FIGUREN

[0015] **Fig.** 1 stellt einen Adreßumsssssetzpuffer und einen Tabellenläufer gemäß dieser Erfindung dar.

[0016] **Fig.** 2 stellt die Schaltung bei einem Ausführungsbeispiel dar, das verwendet wird, um die Gültig-Bits von CAM-Einträgen zu invalidieren.

[0017] **Fig.** 3 gibt die Zeitsteuerung für eine EINSETZEN-Operation an, die von dem TLB bei einem Ausführungsbeispiel ausgeführt wird.

[0018] **Fig.** 4 gibt die Struktur eines Indexknotens eines B-Baums an, der bei einem Ausführungsbeispiel eine Übersetzungstabelle implementiert.

[0019] **Fig.** 5 gibt die Struktur eines Blattknotens eines B-Baums an, der bei einem Ausführungsbeispiel eine Übersetzungstabelle implementiert.

[0020] **Fig.** 6 gibt ein Flussdiagramm an, das das Verarbeiten darstellt, das von einem Tabellenläufer ausgeführt wird, während eine Übersetzungstabelle, die als ein B-Baum implementiert ist, bei einem Ausführungsbeispiel durchsucht wird.

[0021] **Fig.** 7 gibt ein Flussdiagramm an, das das Verarbeiten durch einen Tabellenläufer eines Indexknotens einer Übersetzungstabelle, die als ein B-Baum implementiert ist, bei einem Ausführungsbeispiel darstellt.

[0022] **Fig.** 8 gibt ein Flussdiagramm an, das das Verarbeiten durch einen Tabellenläufer eines Blattknotens einer Übersetzungstabelle, die als ein B-Baum implementiert ist, bei einem Ausführungsbeispiel darstellt.

[0023] **Fig.** 9 gibt ein Blockdiagramm eines Mehrfachtrefferdetektors bei einem Ausführungsbeispiel an, bestehend aus einem Codierer und einem Mehrfachtrefferdetektor.

[0024] **Fig.** 10 gibt einen Codierer an, der bei einem Ausführungsbeispiel in einem Mehrfachtrefferdetektor enthalten ist.

[0025] **Fig.** 11 gibt einen Mehrfachtrefferüberprüfer an, der bei einem Ausführungsbeispiel in einem Mehrfachtrefferdetektor enthalten ist.

GENAUE BESCHREIBUNG DER BEVORZUGTEN AUSFÜHRUNGSBEI-SPIELE

Überblick über den Übersetzungsprozess

[0026] Ein Ausführungsbeispiel eines TLBs gemäß der vorliegenden Erfindung ist in der **Fig.** 1 dargestellt. Der TLB **101** enthält einen Assoziativspeicher (CAM) **102** und einen statischen Direktzugriffsspeicher (SRAM) **103**. Der CAM **102** und SRAM **103** enthalten jeweils **128** adressierbare Elemente (nachfolgend bezeichnet als "Einträge"). Jeder Eintrag des CAMs **102** enthält ein Gültig-Bit-Feld. Wenn das Gültig-Bit für einen speziellen Eintrag des CAMs **102** aufgestellt ist, dann repräsentiert der Eintrag zusammen mit seinem entsprechenden Eintrag im SRAM **103** eine gültige Übersetzung. Andernfalls ist der speziel-

le Eintrag zu ignorieren. Jeder Eintrag des CAMs **102** enthält auch ein Eingangsadressenfeld, dessen übersetzte Adresse in dem entsprechenden Eintrag des SRAMs **103** gespeichert ist. (Bei einigen Ausführungsbeispielen werden Daten, die zum Verarbeiten der übersetzten Adresse verwendet werden, statt der übersetzten Adresse selbst im SRAM **103** gespeichert.)

[0027] Eine Signal auf einer Leitung **104**, das eine Eingangsadresse (X) angibt, deren Übersetzung gewünscht wird (nachfolgend die "spezifizierte Adresse"), wird vom TLB **101** empfangen. Die Inhalte jedes Eintrags des CAMs **102** werden mit der spezifizierten Eingangsadresse verglichen. Diese Aufgabe, die als eine MATCH-Operation bezeichnet wird, wird unten genauer beschrieben.

[0028] Wenn genau ein gültiger Eintrag des CAMs **102** die spezifizierte Eingangsadresse speichert, dann wird die Ausgangsadresse, die die Übersetzung der spezifizierten Eingangsadresse darstellt, von den Inhalten des entsprechenden Eintrags des SRAMs **103** bestimmt. Wenn zwei oder mehr gültige Einträge des CAMs **102** die spezifizierte Eingangsadresse speichern, dann werden diese Einträge invalidiert über eine INVALIDIEREN-Operation, die unten genauer beschrieben ist.

[0029] Eine Übersetzungstabelle **116** (die in einem Ausführungsbeispiel durch eine B-Baum-Datenstruktur implementiert ist), die im Speicher **106** gespeichert ist, wird durch einen Tabellenläufer **115** nach einer Übersetzung für die spezifizierte Eingangsadresse durchsucht, wenn mehr als einer oder keiner der gültigen Einträge des CAMs **102** die spezifizierte Eingangsadresse speichert. Der Tabellenläufer **115** kann, wie unten genauer erörtert ist, von der Übersetzungstabelle **116** nicht nur eine Übersetzung für die spezifizierte Eingangsadresse sondern möglicherweise eine oder mehrere Übersetzungen für andere Eingangsadressen wiedererlangen.

[0030] Jede Übersetzung, die von dem Tabellenläufer **115** wiedererlangt wird, wird in einen Eintrag des CAMs **102** und entsprechenden Eintrag des SRAMs **103** über eine EINSETZEN-Operation eingesetzt, die unten genauer beschrieben ist. Um ein Erzeugen einer Duplikatübersetzung zu vermeiden, wenn in den TLB **101** eine Übersetzung für eine spezielle Eingangsadresse eingesetzt wird, die von der Übersetzungstabelle **116** wiedererlangt wurde, wird eine MATCH-Operation ausgeführt, um zu bestimmen, ob eine Übersetzung für jene spezielle Adresse bereits in einem Eintrag des TLBs **101** gespeichert ist. Wenn genau ein Eintrag des TLBs **101** eine Übersetzung für jene spezielle Adresse speichert, dann wird die Übersetzung für jene spezielle Adresse, die von dem Tabellenläufer **115** von der Übersetzungstabelle **116** wiedererlangt wurde, in jenen genau einen Eintrag eingesetzt, wodurch die Erzeugung von Mehrfach-TLB-Einträgen für die spezielle Adresse vermieden wird. Andernfalls wird der TLB-Eintrag, in welchen die Übersetzung, die von dem Tabellenläufer **115** wiedererlangt wurde, eingesetzt wird, durch die Ersetzpolitik für den TLB **101** bestimmt.

Operationen, die von CAM **102** unterstützt werden

[0031] Der TLB **101** unterstützt mehrere Operationen, einschließlich MATCH, SCHREIBEN, EINSETZEN und INVALIDIEREN. Jede dieser vier Operationen wird unten beschrieben.

TLB-MATCH-Operation

[0032] Eine MATCH-Operation wird in zwei Situationen ausgeführt. In der ersten Situation wird eine MATCH-Operation ausgeführt, wenn der TLB **101** auf der Leitung **104** ein Signal empfängt, das eine Eingangsadresse angibt, deren Übersetzung gewünscht wird. In dieser Situation wählt ein MUX **107** das Signal auf der Leitung **104** aus. Wenn die MATCH-Operation in dieser ersten Situation genau einen gültigen passenden Eintrag des CAMs **102** findet, dann wird die gewünschte Ausgangsadresse von dem entsprechenden Eintrag des SRAMs **103** wiedererlangt. Andernfalls muss der Tabellenläufer **115** die Übersetzungstabelle **116** durchsuchen, die im Speicher **106** gespeichert ist, nach einer Übersetzung für die Eingangsadresse, die durch das Signal auf der Leitung **104** angegeben ist.

[0033] In der zweiten Situation wird eine MATCH-Operation ausgeführt, wenn der TLB **101** auf einer Leitung **105** ein Signal empfängt, das eine Eingangsadresse angibt, deren Übersetzung von der Übersetzungstabelle **116** durch den Tabellenläufer **115** wiedererlangt wurde. In dieser Situation wählt der MUX **107** das Signal auf der Leitung **105**.

[0034] Wenn das Signal auf der Leitung **121**, das von dem CAM-Matchfreigabeport **122** empfangen wird, aufgestellt wird, dann tritt eine MATCH-Operation auf, während der das Eingangsadressenfeld jedes Eintrages des CAMs **102** gleichzeitig mit der Eingangsadresse (nachfolgend "X") verglichen wird, die durch das Signal auf der Leitung **119** angegeben ist, das von dem CAM-Matchport **109** empfangen wurde.

[0035] Matchleitungen **112**, MATCH[127 : 0] (eine für jeden Eintrag des CAMs **102**), sind an den CAM **102** und einen Mehrfachtrefferdetektor (MHD) **150** gekoppelt. Wenn das Eingangsadressenfeld des i-ten Eintrags des CAMs **102**, CAM(i), gleich ist mit ("matcht") X, und das Gültig-Bit von CAM (i) aufgestellt ist, dann wird das Signal auf der i-ten Matchleitung, MATCH(i), aufgestellt. Bei einigen Ausführungsbeispielen wird auch ein Prozess ID, der dem Prozess entspricht, der die virtuelle Adresse erzeugte, die gegenwärtig von der MMU übersetzt wird, dem CAM **102** zugeführt und muss gematcht werden, zusätzlich zur Eingangsadresse, mit einem entsprechenden Feld in einem Eintrag des CAMs **102**. Solche Ausführungsbeispiele können verwendet werden, wo der Übersetzungsschritt entsprechend CAM **102** prozess-id-abhängig ist, um zu vermeiden, dass

die gesamten Inhalte des CAMs **102** nach einem Kontextschalten, d. h. wenn ein unterschiedlicher Prozess zu laufen beginnt, invalidiert werden müssen.

[0036] Der MHD **150** bestimmt, ob null, eines oder mehrere der Signale auf den Matchleitungen **112** aufgestellt sind und setzt zwei Signale MHIT und HIT, die auf Leitungen **113** bzw. **114** übertragen werden. Wenn das Signal auf genau einer der Matchleitungen **112** aufgestellt ist, dann stellt MHD **150** das HIT-Signal auf. Der MHD **150** stellt das MHIT-Signal auf, wenn die Signale auf mehr als einer oder keiner der Matchleitungen **112** aufgestellt sind.

[0037] Bei einem Ausführungsbeispiel hat der MHD **150** die Struktur, die in der **Fig.** 9 dargestellt ist. Ein Codierer **901** codiert die **128** binären Signale, die auf den Matchleitungen **112** befördert werden, in zwei 7-Bit-Signale HAD[6 : 0] und HADN[6 : 0], die auf den Leitungen **903** bzw. **904** befördert werden. Ein Mehrfachtrefferüberprüfer **902** stellt das HIT-Signal auf, wenn HAD und HADN bitweise Komplemente voneinander sind. Andernfalls stellt der Mehrfachtrefferüberprüfer das MHIT-Signal auf. Bei einem Ausführungsbeispiel sind der Codierer **901** und der Mehrfachtrefferüberprüfer **902** die Schaltungen, die in den **Fig.** 10 bzw. 11 dargestellt sind.

[0038] In der **Fig.** 10 ist jedes der Signale auf den Matchleitungen **112**, d. h. MATCH [127 : 0], verbunden mit zwei jeweiligen Sätzen von 7 MOSFET-Transistoren. Die Stellen von Nullen in der 7-Bit-Binärwiedergabe für i, wobei i zwischen 0 und 127, eingeschlossen, ist, bestimmt die Glieder des ersten Satzes von 7 MOSFETs, die mit MATCH[i] verbunden sind, deren Gates verbunden sind mit MATCH[i]. Zum Beispiel ist, wenn i = 1, die Binärwiedergabe für i 0000001. Somit ist MATCH[i] verbunden mit den Gates von MOSFETs **1001**, **1002**, **1003**, **1004**, **1005**, **1006** aber nicht **1007**.

[0039] Die Stellen von Einsen in der 7-Bit-Binärwiedergabe für i, wobei i zwischen 0 und 127, eingeschlossen, ist, bestimmt die Glieder des ersten Satzes von 7 MOSFETs, die mit MATCH[i] verbunden sind, deren Gates verbunden sind mit MATCH[i]. Zum Beispiel ist, wenn i = 1, die Binärwiedergabe für i 0000001. Somit ist MATCH[i] verbunden mit dem Gate von MOSFET **10014** aber nicht mit den Gates der MOSFETs **1008**, **1009**, **1010**, **1011**, **1012**, **1013** aber nicht **1014**.

[0040] Ein Tabellenläufer **115** (siehe **Fig.** 1) empfängt die MHIT- und HIT-Signale auf Leitungen **113** bzw. **114**. In dem Fall von keinen oder mehrfachen matchenden Einträgen im CAM **102** nach einer MATCH-Operation für eine Eingangsadresse, die auf der Leitung **104** empfangen wurde, durchsucht der Tabellenläufer **115** nach einem Detektieren des Aufstellens des MHIT-Signals auf der Leitung **113** die Übersetzungstabelle (gehalten im Speicher **106**) nach einer Übersetzung für die Eingangsadresse X. Dieses Durchsuchen durch den Tabellenläufer ist genauer unten beschrieben. Andererseits wird dann, wenn genau eine der Matchleitungen **112**, MATCH(i), aufgestellt wird als ein Ergebnis der MATCH-Operation, die oben angegebene Durchsuchung des Tabellenläufers nicht initiiert und die Ausgangsadresse, die die Übersetzung der Eingangsadresse X wiedergibt, wird wiedererlangt von dem i-ten Eintrag des SRAMs **103**, SRAM(i).

[0041] Die Zeitsteuerung einer MATCH-Operation ist folgendermaßen. Im ersten Zyklus gibt das Signal auf einer Leitung **108** eine Eingangsadresse X an, die gematcht werden soll. Auch wird in dem ersten Zyklus das Matchfreigabesignal auf einer Leitung **117** aufgestellt. In dem zweiten Zyklus wird das Signal auf der Leitung **108** durch einen Speicher **118** gespeichert und somit gibt das Signal auf der Leitung **119**, das am CAM-Matchport **109** empfangen wurde, die Eingangsadresse X an. Auch im zweiten Zyklus wird das Matchfreigabesignal auf einer Leitung **117** gespeichert durch den Speicher **120** und wird somit das Signal auf einer Leitung **121**, das von einem CRM-Matchfreigabeport **122** empfangen wird, aufgestellt.

[0042] Das Aufstellen der Signale, die auf null, einer oder mehreren Matchleitungen **112** (entsprechend Matcheinträgen des CAMs **102**) übertragen werden, und das Setzen durch den MHD **150** von Signalen MHIT und HIT, die auf Leitungen **113** bzw. 114 übertragen werden, wird erzielt innerhalb der ersten Hälfte des zweiten Zyklus der MATCH-Operation.

[0043] Wenn das Signal auf genau einer der Matchleitungen **112**, MATCH(i), während der ersten Hälfte des zweiten Zyklus einer MATCH-Operation aufgestellt wird, dann werden die Signale auf den Matchleitungen **112** durch den MUX **123** aufgrund des Aufstellens des HIT-Signals ausgewählt. Genau eine der 128 Ausgangsleitungen **173** des MUX **123** wird am Ende der ersten Hälfte des zweiten Zyklus aufgestellt. Jede Ausgangsleitung **173** ist mit einer Eingangsleitung eines jeweiligen der 128 UND-Gates **171** verbunden. TLB-Wortauswahlleitungen **124** sind die jeweiligen Ausgangsleitungen der UND-Gates **171**. Das Taktsignal ist an einem invertierten Eingang jedes der 128 UND-Gates **171** vorgesehen. Somit wird während der zweiten Hälfte des zweiten Zyklus genau eines der Signale, die auf den TLB-Wortauswahlleitungen **124** übertragen werden, WL(i), aufgestellt.

[0044] Wenn die MATCH-Operation für eine Eingangsadresse ausgeführt wurde, die auf der Leitung **104** zugeführt wurde, dann werden die Inhalte von CAM (i) und SRAM(i) am CAM-Leseport **125** bzw. SRAM-Leseport **126** verfügbar gemacht. Die Inhalte von SRAM(i) enthalten die gewünschte Ausgangsadresse entsprechend der Eingangsadresse X. In einigen Ausführungsbeispielen kann ein Feld in SRAM(i) verwendet werden, um eine Paritätsüberprüfung an den Inhalten von CAM(i) auszuführen. Wenn andererseits die MATCH-Operation für eine Eingangsadresse ausgeführt wurde, die auf der Leitung **105** zugeführt wurde, dann werden die Daten, die durch Si-

gnale **155** und **132** (d. h. eine Übersetzung, die von der Übersetzungstabelle **116** durch den Tabellenläufer **115** wiedererlangt wurde) angegeben sind, in CAM(i) bzw. SRAM(i) geschrieben.

TLB-INVALIDIEREN-Operation

[0045] In dem Fall von Mehrfachmatcheinträgen im CAM **102** nach einer MATCH-Operation für eine Eingangsadresse, die auf der Leitung **104** empfangen wurde, setzt eine Logik im TLB **101** die Gültig-Bits der Matcheinträge zurück. In einem Ausführungsbeispiel ist die Logikschaltung, die Gültig-Bits nach der Detektion von Mehrfachmatcheinträgen zurückstellt, wie in der **Fig.** 2 gezeigt ist. Das Gültig-Bit jedes Eintrages des CAMs **102** ist durch eine Standard-6-Transistor-RAM-Zelle implementiert. Zum Beispiel wird der Wert des Gültig-Bits für CAM(0) gespeichert in einer RAM-Zelle **201** als das Signal auf einer Leitung **202**.

[0046] Das Zurückstellen des Gültig-Bits für CAM(0) nach dem Auftreten von Mehrfachmatcheinträgen (einschließlich CAM(0)) wird folgendermaßen erzielt. Nach dem Auftreten von Mehrfachmatcheinträgen werden mehr als eines der Signale auf Matchleitungen **112**, einschließlich MATCH[0], das auf der Eingangsleitung **204** des UND-Gates **203** befördert wird, aufgestellt. Als Antwort stellt der MHD **150** das MHIT-Signal auf der Leitung **113** auf. Als ein Ergebnis des Aufstellens des MHIT-Signals wird das Signal INV_MATCH auf der Eingangsleitung **205** des UND-Gates **203** aufgestellt. Somit wird das Signal einer Ausgangsleitung **206** des UND-Gates **203** aufgestellt. Die Ausgangsleitung **206** ist an das Gate eines MOSFETs **208** gekoppelt. Die Quelle des MOSFETs **208** ist an die Leitung **202** angeschlossen, und die Drain des MOSFETs **208** ist an eine Erdungsquelle **207** angeschlossen. Das Aufstellen des Signals auf der Leitung **206** schaltet den MOSFET **208** ein und somit wird das Signal auf der Leitung **202** mit Erde verbunden, wodurch das Gültig-Bit für CAM(0) zurückgestellt wird. Das Zurückstellen der Gültig-Bits von anderen Einträgen des CAMs wird ähnlich erzielt.

TLB-SCHREIBEN-Operation

[0047] In einer SCHREIBEN-Operation werden die Daten, die durch die Signale auf Leitungen **155** und **132** angegeben sind, in einen Eintrag des CAMs **102** bzw. entsprechenden Eintrag des SRAMs **103** geschrieben. Die SCHREIBEN-Operation wird in zwei Situationen ausgeführt.

[0048] In der ersten Situation wird die SCHREIBEN-Operation spezifiziert durch einen Befehl, der von der CPU ausgeführt wird. In dieser Situation führt die CPU ein Signal auf einer Leitung **138**, das das Element des CAMs **102** und SRAMs **103** angibt, in welche das Schreiben erfolgen soll, ein Signal auf einer Leitung **151**, das die Daten angibt, die in den CAM **102** geschrieben werden sollen, und ein Signal auf einer Leitung **157** zu, das Daten angibt, die in den SRAM **103** geschrieben werden sollen. MUXes **123**, **152** und **156** wählen die Signale auf Leitungen **140**, **151** bzw. **157** aus.

[0049] In der zweiten Situation werden die Daten, die in den CAM **102** und SRAM **103** geschrieben werden sollen, von der Übersetzungstabelle **116** wiedererlangt und durch den Tabellenläufer **115** auf Leitungen **105** bzw. **129** zugeführt. Ein Signal auf einer Leitung **138** gibt das Element des CAMs **102** und SRAMs **103** an, in welches das Schreiben erfolgen soll, und ist entsprechend der TLB-Ersetzpolitik eingestellt. MUXes **123**, **152** und **156** wählen die Signale auf Leitungen **140**, **105** bzw. **129** aus.

[0050] Die Zeitsteuerung einer SCHREIBEN-Operation ist folgendermaßen. In dem ersten Zyklus wird ein Schreibfreigabesignal auf einer Leitung **130** aufgestellt. In dem zweiten Zyklus wird das Schreibfreigabesignal auf der Leitung **130** durch einen Speicher **134** gespeichert und wird somit durch einen CAM-Schreibfreigabeport **136** und SRAM-Schreibfreigabeport **137** wiedererlangt. Ebenfalls während des zweiten Zyklus werden die Ausgaben von MUXes **152** und **156** durch Speicher **153** bzw. **131** gespeichert und werden somit durch einen CAM-Schreibport **110** bzw. SRAM-Schreibport **133** wiedererlangt. Während der ersten Hälfte des zweiten Zyklus wird die Adresse, die durch das Signal auf einer Leitung **138** angegeben wird, durch einen Adressendecoder **139** decodiert. Durch das Beginnen der zweiten Hälfte des zweiten Zyklus wird eine der Adressendecoderleitungen **140** aufgestellt und wird die entsprechende Ausgangsleitung des MUX **123** aufgestellt. Während der zweiten Hälfte des zweiten Zyklus wird eine entsprechende Wortauswahlleitung **124** aufgestellt (dieses Aufstellen wird ausgelöst durch die abfallende Flanke des Taktsignals, das an einem invertierten Eingang jedes der 128 UND-Gates **171** angeschlossen ist) und werden die Daten am CAM-Schreibport **110** und SRAM-Schreibport **133** in den Eintrag des CAMs **102** bzw. entsprechenden Eintrag des SRAMs **103** geschrieben entsprechend der aufgestellten Wortauswahlleitung **124**.

TLB-EINSETZEN-Operation

[0051] Wie oben diskutiert wurde, sucht dann, wenn eine einheitlich gültige Übersetzung für eine Eingangsadresse X, die durch das Signal auf der Leitung **104** angegeben ist, in dem TLB **101** nicht vorhanden ist, der Tabellenläufer **115** nach der gewünschten Übersetzung in der Übersetzungstabelle, einer Datenstruktur, die im Speicher **106** gehalten wird. Wie genauer unten beschrieben ist, könnte der Tabellenläufer **115** mehrere unangeforderte Übersetzungen als ein Ergebnis dieser Suche wiedererlangen. Jede der Übersetzungen, die vom Speicher **106** durch den Tabellenläufer **115** wiedererlangt wird, wird über eine EINSETZEN-Operation in den TLB **101** eingesetzt.

Es gibt eine Möglichkeit, dass eine wiedererlangte aber unangeforderte Übersetzung bereits in dem TLB **101** existiert. Die EINSETZEN-Operation ist gestaltet, um ein Erzeugen mehrerer Einträge für dieselbe Eingangsadresse zu vermeiden, ohne eine Erhöhung bei der Zeit zu erleiden, die benötigt wird, um jene wiedererlangten Übersetzungen einzusetzen, die noch nicht in dem TLB **101** vorhanden sind.

[0052] Die Zeitsteuerung einer EINSETZEN-Operation in einem Ausführungsbeispiel ist, wie in der **Fig.** 3 dargestellt ist. In dem ersten Zyklus erzeugt der Speicher **106** ein Signal auf Leitungen **127A** und **127B**, das eine Eingangsadresse Y, die in einen Eintrag des CAMs **102** einzusetzen ist, bzw. eine entsprechende Ausgangsadresse (oder in einigen Ausführungsbeispielen Ausgangsdaten, von welchen eine Ausgangsadresse berechnet wird) angibt, die in einen entsprechenden Eintrag des SRAMs **103** einzusetzen ist. Zusammen bilden die Signale auf den Leitungen **127A** und **127B** eine Übersetzung, die in den TLB **101** einzusetzen ist. (In einem Ausführungsbeispiel kann der Speicher nur acht Bytes pro Zyklus übertragen, und somit wird das Signal, das einen CAM-Teil der Übersetzung (8 Bytes) wiedergibt, in einem Zyklus übertragen und wird das Signal, das den SRAM-Teil der Übersetzung wiedergibt, über die folgenden zwei Zyklen übertragen.)

[0053] In dem zweiten Zyklus speichern Puffer **128A** und **128B** die in dem Tabellenläufer **115** enthalten sind, die Signale auf den Leitungen **127A** bzw. **127B**, wodurch zwei Signale auf Leitungen **105** und **129** erzeugt werden. Dann führt der TLB **101** eine EINSETZEN-Operation aus, MUXes **107** und **152** wählen das Signal auf der Leitung **105**, und als ein Ergebnis geben die Signale auf Leitungen **108** und **154** auch die Eingansadresse Y während des zweiten Zyklus an. Wenn der TLB **101** eine EINSETZEN-Operation ausführt, wählt ein MUX **156** das Signal auf der Leitung **129**, und als ein Ergebnis gibt das Signal auf der Leitung **158** auch eine Ausgangsadresse entsprechend der Eingangsadresse Y während des zweiten Zyklus an.

[0054] Weiterhin werden in dem zweiten Zyklus die Matchfreigabe- und Schreibfreigabesignale auf Leitungen **117** bzw. **130** aufgestellt. Somit ist eine EINSETZEN-Operation eine Kombination einer MATCH-Operation und einer SCHREIBEN-Operation. Wie unten genauer erläutert wird, vermeidet die Initiierung der MATCH- und SCHREIBEN-Operationen in demselben Zyklus (im Gegensatz zum Ausführen der MATCH-Operation zuerst und dann Entscheiden, ob eine SCHREIBEN-Operation auf der Basis der Ergebnisse der MATCH-Operation durchzuführen ist oder nicht) durch das Aufstellen der Matchfreigabe- und Schreibfreigabesignale auf Leitungen **117** bzw. **130** eine Zunahme bei der Zeit, die erforderlich ist, um eine EINSETZEN-Operation auszuführen.

[0055] In dem dritten Zyklus speichern Speicher **118** und **153** die Signale auf Leitungen **108** bzw. **154**, und somit geben die Signale auf Leitungen **119** und **155**, die vom CAM-Schreibport **110** bzw. CAM-Matchport **109** empfangen werden, die Eingangsadresse Y an. Weiterhin speichert in dem dritten Zyklus ein Speicher **131** das Signal auf der Leitung **158**, und somit gibt das Signal auf der Leitung **132**, das vom SRAM-Schreibport **133** empfangen wird, eine Ausgangsadresse entsprechend der Eingangsadresse Y an. In der ersten Hälfte des dritten Zyklus wird das Matchfreigabesignal auf der Leitung **117** von dem Speicher **120** gespeichert, und somit wird das Signal, das vom CAM-Matchfreigabeport **122** empfangen wird, auf der Leitung **121** aufgestellt. Als ein Ergebnis des Aufstellens des Signals auf der Leitung **121** tritt das Folgende während der ersten Hälfte des dritten Zyklus wie durch eine gewöhnliche MATCH-Operation auf: 1) Eingangsadresse, Y, wird gegenüber den Eingangsadressenfeldern jedes Eintrages des CAMs **102** abgestimmt; 2) null, eines oder mehrere der Signale, die auf Matchleitungen **112** übertragen werden, werden aufgestellt; 3) Mehrfachtrefferdetektor **150** bestimmt, wenn die Eingangsadresse Y mit null, einem oder mehreren gültigen Einträgen im CAM **102** matcht, und 4) MHD **150** setzt die MHIT- und HIT-Signale auf Leitungen **113** bzw. **114**.

[0056] Die Adressensignale auf Leitungen **138** werden durch einen Adressendecoder **139** in dem ersten Halbzyklus des dritten Zyklus decodiert und geben an, welcher Eintrag des TLBs **101** verwendet wird, um die Übersetzung vom Speicher einzusetzen, ohne dass es bereits genau einen Eintrag im CAM **102** gibt, der mit der Eingangsadresse Y matcht. Die Ersetzungspolitik für den TLB **101** bestimmt die Weise, in welcher die Adressensignale auf den Leitungen **138** während einer EINSETZEN-Operation gesetzt werden. In einem Ausführungsbeispiel, wobei FIFO-(zuerst rein, zuerst raus) Ersetzung eingesetzt wird, werden die Adressensignale auf den Leitungen **138** anfänglich gesetzt, um auf CAM(0) zu weisen. Nach jeder EINSETZEN-Operation werden Adressensignale auf den Leitungen **138** geändert, um auf CAM ((i + 1) mod **128**) zu weisen, wenn sie gegenwärtig auf CAM(i) weisen, bis genau ein CAM-Eintrag die Eingangsadresse trifft, die in den CAM **102** einzusetzen ist, in welchem Fall die Adressensignale auf den Leitungen **138** nicht geändert werden.

[0057] Wenn der MHD **150** bestimmt, dass die Eingangsadresse Y mit mehreren oder keinen Einträgen des CAMs **102** matcht und daher MHD **150** das HIT-Signal auf der Leitung **114** (welches die Auswahlleitung des MUX **123** betreibt) in der ersten Hälfte des dritten Zyklus zurücksetzt, dann werden die Signale auf Adressendecoderleitungen **140** durch den MUX **123** ausgewählt und die CRM-Wortauswahlleitung **124**, die der aufgestellten Adressendecoderleitung **140** entspricht, wird in der zweiten Hälfte des dritten Zyklus aufgestellt (dieses Aufstellen wird ausgelöst durch die fallende Flanke des Taktsignals, das an einen invertierten Eingang jedes der 128 UND-Gates **171** angeschlossen ist).

[0058] Andernfalls, d. h. in dem Fall eines einzelnen

Eintrags des CAMs **102**, der mit der Eingangsadresse Y matcht, setzt der MHD **150** das HIT-Signal auf der Leitung **114** in der ersten Hälfte des dritten Zyklus zurück und werden als ein Ergebnis Matchleitungen **112** durch den MUX **123** ausgewählt. Während der zweiten Hälfte des dritten Zyklus wird die Wortauswahlleitung **124**, die der aufgestellten Matchleitung **112** entspricht, aufgestellt (dieses Aufstellen wird ausgelöst durch die fallende Flanke des Taktsignals, das an einen invertierten Eingang jedes der **128** UND-Gates **171** angeschlossen ist).

[0059] Weiterhin wird in dem dritten Zyklus das Schreibfreigabesignal auf der Leitung **130** durch den Speicher **134** gespeichert, und somit wird das Signal auf der Leitung **135**, das von dem CAM-Schreibfreigabeport **136** und SRAM-Schreibfreigabeport **137** empfangen wird, aufgestellt. Somit werden in dem dritten Zyklus die Übersetzungsdaten, die durch die Signale auf den Leitungen **119** (d. h. Eingangsadresse) und **132** (Ausgangsadresse) angegeben sind, in den Eintrag von CAM **102** und entsprechenden Eintrag SRAM **103** geschrieben, ausgewählt durch TLB-Wortauswahlleitungen **124**.

[0060] Die obige EINSETZEN-Operation verhindert das Erzeugen von Mehrfachmatcheinträgen im TLB **101** in dem Fall eines bereits in dem TLB **101** vorhandenen einzelnen Matcheintrages durch Betreiben von TLB-Wortauswahlleitungen **124** mit Matchleitungen **112**, wodurch über den Matcheintrag die Übersetzung gelegt wird, die vom Speicher **106** durch den Tabellenläufer **115** geholt wird, im Gegensatz zum Eintragen der geholten Übersetzung in einen anderen TLB-Eintrag, d. h. den, der durch die Adressensignale auf Leitungen **138** angegeben ist. Andererseits erhöht das Matchen, das während eines EINSET-ZENs ausgeführt wird, um die Erzeugung von Mehrfachmatcheinträgen zu verhindern, die Zeit nicht, die erforderlich ist, um die Übersetzungsdaten, die vom Speicher **106** geholt wurden, in den TLB-Eintrag einzusetzen, der durch die Adressen-signale auf den Leitungen **138** angegeben ist, in dem Fall von keinen oder Mehrfachmatch-CAM-Einträgen.

[0061] Dies liegt daran, dass Schreib- und Matchoperationen gleichzeitig durch das Aufstellen des Matchfreigabe- und des Schreibfreigabesignals auf Leitungen **117** bzw. **130** in demselben Zyklus gestartet werden, und daran, dass das Decodieren der Adressensignale auf den Leitungen **138**, die den Eintrag des TLBs **101** angeben, in welchen die Übersetzung, die einzusetzen ist, platziert wird, wenn es nicht genau einen Matcheintrag im TLB **101** gibt, gleichzeitig auftritt (während er ersten Hälfte des dritten Zyklus) mit dem Matchen der Eingangsadresse, die mit der Übersetzung verbunden ist, gegenüber den Einträgen des CAMs **102**.

[0062] Wenn der Speicher **106** den Tabellenläufer **115** mit einer Übersetzung pro Zyklus versorgen kann, dann kann die oben beschriebene EINSET-ZEN-Operation eine effektive Einsetzrate von ungefähr einer Übersetzung pro Zyklus unterstützen, da die Übersetzungen längs Leitungen **127A**, **105**/**154** und **155** (zum Schreiben in CAM **102**), längs Leitungen **127A**, **105**/**108** und **119** (zum Matchen mit CAM **102**) und längs Leitungen **127B**, **129**/**158** und **132** (zum Einsetzen in SRAM **103**) bevorratet werden können. Zusätzlich brauchen Puffer **128A** und **128B** im Tabellenläufer **115** nur groß genug sein, um eine Übersetzung zu halten, da der Tabellenläufer **115** Übersetzungen auf Leitungen **105** und **129** hinausschicken kann, so schnell wie der Tabellenläufer **115** Übersetzungen vom Speicher **106** empfängt.

[0063] Wie oben diskutiert wurde, treten das tatsächliche Matchen und Schreiben in einer EINSET-ZEN-Operation für den TLB **101** in demselben Zyklus auf. Andererseits könnte ein TLB, der Einsetzungen durch zuerst Matchen und dann Schreiben in einem nachfolgenden Zyklus (in Abhängigkeit von den Ergebnissen des Matchens) nicht eine effektive Rate von Einsetzungen unterstützen, die eine Übersetzung pro zwei Zyklen übersteigt, unter der Annahme, dass der TLB außer Stande ist, eine zweite Eingangsadresse zu matchen, während eine erste Eingangsadresse geschrieben wird, die in dem vorhergehenden Zyklus gematcht wurde. Auch würde Pufferraum, der ausreichend ist, um N/2 Übersetzungen zu speichern, in einem solchen TLB erforderlich sein, wobei N die maximale Anzahl von Übersetzungen ist, die vom Speicher geliefert werden, während jeglicher Durchsuchung der Übersetzungstabelle, wenn Übersetzungen von dem Speicher zum Tabellenläufer mit der Rate von einer Übersetzung pro Zyklus geliefert werden.

[0064] Zusätzlich mangelt es einer EINSET-ZEN-Operation, die gestaltet ist, um ein Matchen vor einem Initiieren des Schreibens der geholten Übersetzung auszuführen, an anderen Nachteilen. Da die maximale effektive Einsetzrate eine Übersetzung pro zwei Zyklen wäre, kann der Tabellenläufer **115** nur eine Übersetzung auf Signalleitungen **105** und **129** alle zwei Zyklen schicken. Somit erfordert der Tabellenläufer einen Puffer, der ungefähr N/2 Übersetzungen speichern kann, wobei N die maximale Anzahl von Übersetzungen ist, die vom Speicher **106** in aufeinanderfolgenden Zyklen geliefert werden.

Hohlen von Übersetzungsdaten nach TLB-Fehlschlag

[0065] Wenn die Übersetzung für eine spezielle Eingangsadresse, X, nicht in dem TLB **101** vorhanden ist, sucht der Tabellenläufer im Speicher **106** (Übersetzungstabelle **116**), die erzeugt und aufrechterhalten wird von dem Betriebssystem, eine Datenstruktur, die Übersetzungen für den gegenwärtig laufenden Prozess enthält. In einem Ausführungsbeispiel ist eine Übersetzungstabelle **116**, die nachfolgend als der "Übersetzungsbaum" bezeichnet wird, als ein B-Baum implementiert. B-Bäume sind wohlbekannte Datenstrukturen und sind definiert und beschrieben auf den Seiten 305–327 in "Design of Database

Structures" von Teorey und Fry (1982, Prentice Hall Inc.), was hiermit durch Bezugnahme aufgenommen ist. Wie darin beschrieben ist, gibt es mehrere Varianten eines B-Baums. Der Ausdruck B-Baum wird hierin verwendet, um jegliche von diesen und vielleicht anderen Variationen des herkömmlichen B-Baums zu bezeichnen. In diesem Ausführungsbeispiel, das unmittelbar unten beschrieben wird, ist eine Übersetzungstabelle **116** durch die Variante des B-Baums implementiert, die als ein B*-Baum bekannt ist.

[0066] Jeder Knoten eine B*-Baums ist entweder ein Blattknoten oder ein Indexknoten. In einem Ausführungsbeispiel ist jeder Übersetzungsbaumknoten von einer Größe gleich der Speichercacheleitungsgröße von 128 Bytes und ist jeder Indexknoten des Übersetzungsbaums von der Form, die in der **Fig.** 4 dargestellt ist. Der Indexknoten **400** besteht abwechselnd aus Knotenzeigern **401** und Schlüsseln **402**. Jeder von acht Knotenzeigern **401** und sieben Schlüsseln **402** ist 8 Bytes lang. Jeder Schlüssel speichert eine Eingangsadresse, die gegenüber X verglichen werden kann, die Eingangsadresse die zu übersetzen ist. Bis zu 6 der letzten Schlüssel in jedem Indexknoten können Null sein. Der Knotenzeiger, der einem Null-Schlüssel folgt, ist ungültig. Die Werte der Nicht-Null-Schlüssel in einem Indexknoten nehmen im Wert von links nach rechts zu. Die letzten 8 Bytes von jedem Indexknoten werden ignoriert.

[0067] Jeder Knotenzeiger **401** enthält die folgenden Felder: ein Knotenadressenfeld, das auf einen weiteren Knoten in dem Übersetzungsbaum weist; ein Knotenadressentypfeld, das den Adressentyp definiert, der in dem Knotenadressenfeld gespeichert ist (siehe Beschreibung unten von einigen möglichen Typen von Knotenadressen); ein Knotentypfeld, das den Knotentyp angibt (Blatt oder Index), auf den durch die Adresse in dem Adressenfeld gezeigt wird, und ein Übersetzungsnummernfeld, das die Nummer von gültigen Übersetzungen angibt, die in dem Blattknoten gespeichert sind, auf den durch die Adresse in dem Knotenadressenfeld gezeigt wird (nur anwendbar, wenn das Knotentypfeld ein 'Blatt' ist).

[0068] Wie oben angegeben ist, kann die Adresse, die in dem Knotenadressenfeld eines Knotenzeigers **401** gespeichert ist, mehrere Formen annehmen. In verschiedenen Ausführungsbeispielen können die folgenden sowie möglicherweise andere Knotenadressentypen vorgesehen sein:

1) Real: Die Knotenadresse repräsentiert eine reale oder physikalische Speicheradresse, die direkt verwendet werden kann, um aus dem Speicher den Übersetzungsbaumknoten wiederzuerlangen, auf den gezeigt wird.

2) Logisch: Die Knotenadresse repräsentiert eine logische Adresse einer Art, die natürlich selbst in eine reale Adresse übersetzt werden muss, bevor der Übersetzungsbaumknoten, auf den gezeigt wird, von dem Speicher wiedererlangt werden kann. In einigen Ausführungsbeispielen wird diese Übersetzung ziemlich schnell erzielt. Zum Beispiel könnte die Knotenadresse einen Seitenversatz innerhalb der Seite repräsentieren, in welcher der Knotenzeiger **401** sitzt, in welchem Fall die entsprechende reale Adresse bloß durch Verketten der realen Seitenadresse, in welcher der Knotenzeiger **401** sitzt, mit der Knotenadresse gebildet wird, die in dem Knotenzeiger **401** gespeichert ist. In einem anderen Ausführungsbeispiel könnte in der MMU ein spezieller Adreßumssetzpuffer vorgesehen sein, um eine logische Adresse, die in dem Knotenadressenfeld des Knotenzeigers **401** gespeichert ist, in eine reale Adresse zu übersetzen.

[0069] In einem Ausführungsbeispiel ist ein Blattknoten des Übersetzungsbaums von der Form, die in der **Fig.** 5 dargestellt ist. Der Blattknoten **500** belegt 128 Bytes (die angenommene Cacheleitungsgröße des Speichers in diesem Ausführungsbeispiel) und speichert fünf Übersetzungsbeschreiber **501** (jeder belegt 24 Bytes). Die letzten 8 Bytes eines Blattknotens werden ignoriert. Jeder von Übersetzungsbeschreibern **501** belegt 24 Bytes und besteht aus einem CAM-Datenteil **502**, der 8 Bytes belegt, und einem SRAM-Datenteil **503**, der 16 Bytes belegt. Der CAM-Datenteil **502** und SRAM-Datenteil **503** enthält Übersetzungsdaten, die in einen Eintrag von CAM **102** (z. B. Eingangsadresse) bzw. den entsprechenden Eintrag SRAM **103** (z. B. Ausgangsadresse) einzusetzen sind. Zusätzlich enthält der CAM-Datenteil **502** ein Gültig-Bit, das angibt, ob sein zugehöriger Übersetzungsbeschreiber gültig ist oder nicht.

[0070] Ein spezieller Register in der MMU wird verwendet, um einen Knotenzeiger, nachfolgend bezeichnet als der "TTRP" (Übersetzungsbaumwurzelzeiger) zu halten, der dasselbe Format hat wie jener von Knotenzeigern, die in Indexknoten des Übersetzungsbaums gespeichert sind, wie oben beschrieben wurde, und deren Knotenadressenfeld eine Adresse enthält, die auf den Wurzelknoten des Übersetzungsbaums für den gegenwärtig laufenden Prozess zeigt. Nach einem Kontextschalten (d. h. wenn ein unterschiedlicher Prozess die Ausführung beginnt), aktualisiert das Betriebssystem die Inhalte des TTRP-Registers, um auf den Wurzelknoten des Übersetzungsbaums für den neu laufenden Prozess zu zeigen.

[0071] Die Schritte, die in einem Durchsuchen des Übersetzungsbaums durch den Tabellenläufer **116** involviert sind, sind durch das Flussdiagramm **600** der **Fig.** 6 dargestellt. Das Verarbeiten durch den Tabellenläufer **116** beginnt im Schritt 601, wo der variable Aktueller_Knoten_Zeiger, der im Speicher gespeichert ist, initialisiert wird auf den Knotenzeiger, der in den TTRP-Register gespeichert ist. Im Verlauf des Verarbeitens, das durch die **Fig.** 6 dargestellt ist, zeigt das Knotenadressenfeld von Aktueller_Knoten_Zeiger auf den gegenwärtig durchsuchten Übersetzungsbaumknoten. Das Verarbeiten geht vom Schritt 601 zu einem Entscheidungsschritt **602**.

[0072] Im Entscheidungsschritt **602** untersucht der Tabellenläufer **615** das Knotentypfeld von Aktueller Knoten-Zeiger. Wenn das Typfeld "Blatt" angibt, dann geht das Verarbeiten vom Schritt 602 zum Schritt 603. Im Schritt 603 verarbeitet der Tabellenläufer **115** den Blattknoten, auf den durch das Knotenadressenfeld von Aktueller Knoten-Zeiger gezeigt wird. Vom Schritt 603 geht das Verarbeiten zum Schritt 604, wo das Verarbeiten endet.

[0073] Wenn der Tabellenläufer 115 im Schritt 602 bestimmt, dass das Knotentypfeld von Aktueller Knoten-Zeiger "Index" ist, dann geht das Verarbeiten vom Schritt 602 nach 605. Im Schritt 605 verarbeitet der Tabellenläufer **115** den Indexknoten, auf den durch das Knotenadressenfeld von Aktueller Knoten-Zeiger gezeigt wird. Durch das Endes des Schrittes 605 hält Aktueller Knoten Zeiger einen Zeiger, der in dem Übersetzungsbaumknoten gespeichert ist, der gerade durchsucht wurde, und dessen Knotenadressenfeld auf den nächsten Übersetzungsbaumknoten weist, der von dem Tabellenläufer **115** bearbeitet werden soll. Vom Schritt 605 geht die Verarbeitung zum Schritt 602.

[0074] Der Schritt 605 von **Fig.** 6 für das Verarbeiten eines Indexknotens des Übersetzungsbaums durch den Tabellenläufer **115** ist genauer durch das Flussdiagramm **700** der **Fig.** 7 dargestellt. Das Verarbeiten beginnt im Schritt 701, in welchem der Tabellenläufer **115** die Speichercacheleitung anfragt beginnend an der realen Adresse, die gespeichert ist in (oder entspricht, in dem Fall einer logischen Knotenadresse) dem Knotenadressenfeld von Aktueller Knoten-Zeiger. Der Speicher **106** nimmt mehrere Zyklen, um das Lesen aufzustellen, und schickt dann in einer Sequenz von nacheinanderfolgenden Zyklen eine feste Anzahl (8 in einem Ausführungsbeispiel) Bytes der angefragten Cacheleitung zum Tabellenläufer **115**, bis alle 128 Bytes der Cacheleitung empfangen wurden. Das Verarbeiten geht vom Schritt 701 zum Schritt 702, während welchem der Tabellenläufer **115** die ersten 8 Bytes (einen Knotenzeiger), die vom Speicher **106** zurückkommen, in Aktueller Knoten Zeiger speichert. Weiterhin wird während des Schrittes 702 eine Zählervariable, i, auf 1 initialisiert. Der Zweck der Zählervariable i ist, sicherzustellen, dass die Schleife, die durch Verarbeitungsschritte **703**–**705** definiert ist, die unten beschrieben sind, nicht öfter als 7 mal ausgeführt wird.

[0075] Das Verarbeiten geht vom Schritt 702 zum Entscheidungsschritt **703**, in welchem die Zählervariable i mit 7 verglichen wird. Wenn i größer als 7 ist, dann geht das Verarbeiten zum Schritt 707, wo das Verarbeiten endet. Wenn i nicht größer als 7 ist, dann geht das Verarbeiten zum Schritt 704, wo der Tabellenläufer **115** den Schlüssel, der in den nächsten 8 Bytes enthalten ist, die von dem Speicher empfangen wurden, mit X vergleicht, der Eingangsadresse, deren Übersetzung erwünscht ist. Wenn X geringer als der Schlüssel ist, dann geht das Verarbeiten zum Schritt 707, wo das Verarbeiten endet. Andererseits geht, wenn X nicht geringer als der Schlüssel ist, dann das Verarbeiten zum Schritt 705, wo der Tabellenläufer **115** die nächsten 8 Bytes, die von dem Speicher (einem Knotenzeiger) empfangen wurden, in Aktueller Knoten-Zeiger speichert. Weiterhin wird im Schritt 705 die Zählervariable i um 1 inkrementiert. Vom Schritt 705 geht das Verarbeiten zum Schritt 703.

[0076] Der Schritt 603 von **Fig.** 6 für das Verarbeiten eines Blattknotens des Übersetzungsbaums durch den Tabellenläufer **115** ist genauer durch das Flussdiagramm **800** von der **Fig.** 8 dargestellt. Das Verarbeiten beginnt im Schritt 801, in welchem der Tabellenläufer **115** die Speichercacheleitung anfragt beginnend an der realen Adresse, die gespeichert ist in (oder entspricht, in dem Fall einer logischen Knotenadresse) dem Knotenadressenfeld von Aktueller_Knoten_Zeiger.

[0077] Der Speicher **106** nimmt mehrere Zyklen, um das Lesen aufzustellen, und schickt dann eine Sequenz von aufeinanderfolgenden Zyklen einer festen Anzahl (8 in einem Ausführungsbeispiel) Bytes der angefragten Cacheleitung zum Tabellenzeiger **115**, bis alle 128 Bytes der Cacheleitung empfangen wurden.

[0078] Ebenfalls ist eine Zählervariable, i, in dem Schritt 801 auf 1 eingestellt. Der Zweck der Zählervariable ist, um sicherzustellen, dass die Schleife, die durch die Schritte 802, 806, 807 und 808 definiert ist, die weiter unten beschrieben sind, nicht mehr als 5 mal ausgeführt wird. Weiterhin wird im Schritt 801 die Nummer von gültigen Übersetzungen, die in dem gegenwärtig verarbeiteten Blattknoten vorhanden sind, wie in dem Übersetzungsnummernadressenfeld von Aktueller_Knoten_Zeiger gespeichert, in der Variablen "num trans" gespeichert. Ebenfalls wird eine boolsche Variable "gefunden", die verwendet wird, um anzugeben, ob die gewünschte Übersetzung lokalisiert wurde oder nicht, im Schritt 801 auf falsch initialisiert.

[0079] Das Verarbeiten geht vom Schritt 801 zum Entscheidungsschritt **802**, in welchem die Zählervariable i mit num trans verglichen wird. Wenn i größer als num trans ist, d. h. die erwartete Nummer von gültigen Übersetzungen, die in dem gegenwärtigen Blattknoten erwartet wurde, dann geht das Verarbeiten vom Entscheidungsschritt 802 zum Entscheidungsschritt **803**, wo der Wert der boolschen Variable "gefunden" untersucht wird. Wenn "gefunden" gleich falsch ist, was angibt, dass die gewünschte Übersetzung nicht gefunden wurde, dann geht das Verarbeiten zum Schritt 809, wo der Tabellenläufer **115** eine Unterbrechung erzeugt, die dem Betriebssystem angibt, dass die gewünschte Übersetzung nicht in der Übersetzungstabelle **116** gefunden wurde, und das Betriebssystem veranlasst, eine geeignete Aktion zu unternehmen. Andernfalls geht das Verarbeiten zum Schritt 804, wo das Verarbeiten endet.

[0080] Wenn i nicht größer als num_trans ist, dann geht das Verarbeiten vom Entscheidungsschritt 802 zum Entscheidungsschritt **806**, wo das Gültig-Bit, das

in den nächsten (d. h. für i = 1, das erste) 24 Bytes (d. h. den nächsten Übersetzungsbeschreiber) der angefragten Cacheleitung durch den Tabellenläufer **115** untersucht wird. Wenn das Gültig-Bit nicht gesetzt ist (wodurch angegeben ist, dass der Übersetzungsbeschreiber nicht gültig ist), geht das Verarbeiten vom Schritt 806 zum Schritt 805, wo der Tabellenläufer **115** eine Unterbrechung erzeugt, um das Betriebssystem zu veranlassen, ein Ausführen zu beginnen, und das Betriebssystem zu informieren, dass das Veranlassen der Unterbrechung die Tatsache war, dass die Nummer von gültigen Übersetzungen, die in dem gegenwärtigen Blattknoten gefunden wurde, kleiner als die Nummer war, die erwartet wurde, d. h. die Nummer, die in der Variablen "num trans" im Schritt 801 gespeichert wurde.

[0081] Wenn das Gültig-Bit gesetzt ist, dann geht das Verarbeiten vom Schritt 806 zum Schritt 807, wo die boolsche Variable "gefunden" auf wahr gesetzt wird, wenn die Eingangsadresse, die in den ersten 8 Bytes des gegenwärtig untersuchten Übersetzungsbeschreibers gespeichert ist (d. h. die 24 Bytes der angefragten Cacheleitung, auf die im Schritt 806 Bezug genommen wurde), gleich ist mit X, der Eingangsadresse, nach deren Übersetzung der Tabellenzeiger **115** sucht. Vom Schritt 807 geht das Verarbeiten zum Schritt 808, wo der Tabellenläufer **115** eine EINSETZEN-Operation im TLB **101** initiiert, um Daten, die enthalten sind in (und/oder in denselben Ausführungsbeispielen berechnet wurden von) den ersten 8 Bytes (Eingangsadresse) und den letzten 16 Bytes (Ausgangsadresse) des Übersetzungsbeschreibers, der von dem Speicher empfangen wurde, der gegenwärtig von dem Tabellenzeiger **115** untersucht wird, in einen Eintrag von CAM **102** bzw. dem entsprechenden Eintrag von SRAM einzusetzen. (Wie oben beschrieben ist, ist die TLB-EINSETZEN-Operation gestaltet, um das Einsetzen eines Duplikateintrages zu verhindern.) Weiterhin wird im Schritt 808 die Zählervariable i um 1 inkrementiert. Vom Schritt 808 geht das Verarbeiten zum Schritt 802. Nachdem der Tabellenzeiger **115** einen Blattknoten des Übersetzungsbaums verarbeitet hat, werden die angefragten Übersetzungsdaten, die einer Eingangsadresse X entsprechen, in TLB **101** eingesetzt (angenommen, sie wurde in dem Blattknoten gefunden; wenn nicht eine Unterbrechung erzeugt wurde, wie oben beschrieben ist), wie auch die Übersetzungsdaten entsprechend bis zu vier anderen Eingangsadressen.

[0082] Die Zusatzzeit, die zum Einsetzen unangeforderter Übersetzungen erforderlich ist, die in demselben Übersetzungsbaumblattknoten gespeichert sind, wie die angefragte Übersetzung, ist klein im Vergleich zur Gesamtzeit, die erforderlich ist, um die angefragte Übersetzung zu lokalisieren. Bei der Referenzlokalität, die von den meisten Programmen gezeigt wird (d. h. wenn ein Programm eine Eingangsadresse X referenziert, ist es wahrscheinlich, in der nahen Zukunft eine Eingangsadresse nahe X zu referenzieren) und der Tatsache, dass ein Blattknoten des Übersetzungsbaums Übersetzungen für nahe lokalisierte Eingangsadressen speichert, sollte ein Einsetzen von unangefragten Übersetzungen die Verfehlungsrate in der TLB herabsetzen und somit die Durchschnittszeit verringern, die erforderlich ist, um eine Adressenübersetzung auszuführen. Im Effekt kann der Aufwand des Ausführens eines Durchsuchens einer Übersetzungstabelle **116** über zwei oder mehr Übersetzungen amortisiert werden.

Patentansprüche

1. Verfahren zur Adressenübersetzung in einem Computersystem, welches Computersystem enthält
einen Speicher (**106**), der eine Tabelle (**116**) speichert, die Übersetzungen hält, und
einen Übersetzungspuffer (**102**, **103**), der in jeweils adressierbaren Elementen davon speichert:
Übersetzungen, die von der Tabelle erlangt wurden und von denen jede eine Eingabeadresse (**502**) und eine Ausgabeadresse (**503**) enthält, die verwendet werden, um eine Ausgabeadresse entsprechend der Eingabeadresse zu bestimmen, und
Indikationen, ob die jeweils gespeicherten Übersetzungen gültig sind,
welcher Übersetzungspuffer einem Zeiger (**138**) zugeordnet ist, der ein adressierbares Element des Übersetzungspuffers angibt, in welches eine nächste Übersetzung, die von der Tabelle erlangt wurde, einzusetzen ist,
welches Verfahren die Schritte enthält:
(A) Empfangen (**104**) einer ersten Eingabeadresse;
(B) Durchsuchen des Übersetzungspuffers (**102**) nach einer Übersetzung, die die erste Eingabeadresse enthält;
(B1) wenn der Schritt des Durchsuchens des Übersetzungspuffers wenigstens zwei gültige Übersetzungen findet, Platzieren von Ungültigkeitsindikationen in den Elementen des Übersetzungspuffers, der die wenigstens zwei Übersetzungen speichert, dann
(C) Durchsuchen der Tabelle nach einer Übersetzung, die die erste Eingabeadresse enthält,
(B2) wenn der Schritt des Durchsuchens des Übersetzungspuffers scheitert, eine gültige Übersetzung zu finden, die die erste Eingabeadresse enthält, dann
(C) Durchsuchen der Tabelle nach einer Übersetzung, die die erste Eingabeadresse enthält;
(D) nach dem Schritt des Durchsuchens der Tabelle
(E) Einsetzen einer Übersetzung, die die erste Eingabeadresse enthält, in den Übersetzungspuffer und
(F) Einsetzen einer Übersetzung, die eine zweite Eingabeadresse enthält, in den Übersetzungspuffer, wobei die zweite Eingabeadresse nicht gleich der ersten Eingabeadresse ist, welche Übersetzung, die die zweite Eingabeadresse enthält, gefunden wurde während des Schrittes des Durchsuchens der Tabelle;
und welcher Schritt des Einsetzens einer Übersetzung, die eine zweite Eingabeadresse enthält, die in

der Tabelle gefunden wurde, ferner enthält die Schritte:
(G) weiteres Durchsuchen des Übersetzungspuffers nach einer Übersetzung, die die zweite Eingabeadresse enthält; und
(G1) wenn der Schritt des Durchsuchens des Übersetzungspuffers nach einer Übersetzung, die die zweite Eingabeadresse enthält, scheitert, eine gültige Übersetzung zu lokalisieren, die die zweite Eingabeadresse enthält, dann
(H1) Einsetzen der Übersetzung, die die zweite Eingabeadresse enthält, die in der Tabelle gefunden wurde, in ein Element des Übersetzungspuffers, das durch den Zeiger angegeben ist;
(G2) wenn der Schritt des Durchsuchens des Übersetzungspuffers nach einer Übersetzung, die die zweite Eingabeadresse enthält, genau eine gültige Übersetzung findet, die die zweite Eingabeadresse enthält, dann
(H2) Einsetzen der Übersetzung, die die zweite Eingabeadresse enthält, die in der Tabelle gefunden wurde, in das Element des Übersetzungspuffers, in welchem die Übersetzung beim Durchsuchen des Übersetzungspuffers gefunden wurde.

2. Verfahren nach Anspruch 1, wobei jede erhaltene Eingabeadresse ein Kennungsfeld enthält und jede Übersetzung, die in dem Übersetzungspuffer gespeichert ist, auch ein Kennungsfeld enthält, wobei eine gespeicherte Übersetzung als eine Übersetzung einer Eingabeadresse nur angesehen wird, wenn die Kennungsfelder der Eingabeadresse und der Übersetzung denselben Wert haben.

3. Verfahren nach Anspruch 1, wobei der Schritt des Durchsuchens des Übersetzungspuffers nach einer Übersetzung, die die zweite Eingabeadresse enthält, eine Dauer des Schrittes des Einsetzens der Übersetzung, die die zweite Eingabeadresse enthält, die in der Tabelle gefunden wurde, in den Übersetzungspuffer nicht verlängert.

4. Verfahren zur Adressenübersetzung nach Anspruch 2, wobei die Übersetzungstabelle nicht in einer Seitentabelle gespeichert ist.

5. Verfahren nach einem der vorhergehenden Ansprüche, wobei die Tabelle eine B-Baum-Datenstruktur ist, die wenigstens einen Knoten enthält.

6. Verfahren nach Anspruch 5, wobei die B-Baum-Datenstruktur Blattknoten und Indexknoten enthält, jeder der Blattknoten wenigstens eine Übersetzung speichert und jeder der Indexknoten wenigstens einen Zeiger speichert, der zu einem anderen Knoten der B-Baum-Datenstruktur zeigt, welches Verfahren ferner enthält, wenn die Tabelle durchsucht wird:
Bestimmen, ob ein Knoten der B-Baum-Datenstruktur, auf die durch das Knotenadressenfeld eines Knotenzeigers gezeigt wird, ein Blattknoten oder ein Indexknoten ist; Wiedererlangen und Verarbeiten des Knotens in Abhängigkeit davon, ob er bestimmt ist, ein Indexknoten oder ein Blattknoten zu sein, wobei in dem Fall eines Blattknotens im Fall des Einsetzens einer Übersetzung davon in den Übersetzungspuffer jede Übersetzung, die in dem betroffenen Blattknoten gespeichert ist, in den Übersetzungspuffer eingesetzt wird.

7. Verfahren nach Anspruch 5 oder 6, wobei die Übersetzung, die die erste Eingabeadresse enthält, und die Übersetzung, die die zweite Eingabeadresse enthält, in demselben Knoten der B-Baum-Datenstruktur gespeichert werden.

8. Verfahren nach Anspruch 5, 6 oder 7, wobei eine Größe von wenigstens einem Knoten der B-Baum-Datenstruktur gleich einer Cache-Leitungsgröße des Speichers ist.

9. Adressenübersetzungsgerät in einem Computersystem, welches Computersystem einen Speicher (**106**) enthält, der eine Tabelle (**116**) speichert, die Übersetzungen hält, wobei jede der Übersetzungen eine Eingabeadresse (**502**) und eine Ausgabeadresse (**503**) enthält, die verwendet werden, um eine Ausgabeadresse entsprechend der Eingabeadresse zu bestimmen, welches Gerät enthält:
einen Übersetzungspuffer (**102**, **103**) der in jeweils adressierbaren Elementen davon Übersetzungen speichert, die von der Tabelle erlangt wurden, wobei jedes Element des Übersetzungspuffers auch ein Gültigkeitsfeld zum Speichern einer Angabe enthält, ob die gespeicherte Übersetzung gültig ist; und
eine Zeigerlogik (**138**, **139**), die operativ mit dem Übersetzungspuffer verbunden ist, wodurch die Zeigerlogik ein Element des Übersetzungspuffers angibt, in welches eine nächste Übersetzung, die von der Tabelle erlangt wurde, einzusetzen ist;
welches Gerät ferner enthält
einen Übersetzungspuffer-Durchsucher (**104**, **107**, **108**, **118**, **119**, **102**), operativ verbunden mit dem Übersetzungspuffer und betreibbar, um ein Durchsuchen des Übersetzungspuffers nach einer Übersetzung auszuführen, die eine spezifizierte erste Eingabeadresse enthält;
einen Übersetzungspuffer-Invalidator, der operativ mit dem Übersetzungspuffer verbunden ist und betreibbar ist, um, wenn der Übersetzungspuffer-Durchsucher gültige Adressen, die die spezifizierte erste Eingabeadresse enthalten, in jedem von wenigstens zwei Elementen des Übersetzungspuffers findet, Ungültigkeitsangaben in den Gültigkeitsfeldern der wenigstens zwei Elemente zu platzieren, und um ein Durchsuchen der Tabelle (116) nach einer Übersetzung zu initiieren, die die spezifizierte erste Eingabeadresse enthält,
einen Tabellenläufer (**116**), der operativ mit dem Speicher (**106**) und dem Übersetzungspuffer (**102**,

103) verbunden ist,
betreibbar, um die Tabelle nach einer Übersetzung zu durchsuchen, die die erste spezifizierte Eingabeadresse enthält, wenn der Übersetzungspuffer-Durchsucher scheitert, eine gültige Übersetzung der spezifizierten ersten Eingabeadresse zu finden, oder wenn ein Durchsuchen der Tabelle von dem Übersetzungspuffer-Invalidator initiiert ist, und
betreibbar, um von der Tabelle eine Übersetzung wieder zu erlangen, die die erste Eingabeadresse und eine weitere Übersetzung enthält, die eine zweite Eingabeadresse enthält, die nicht gleich der spezifizierten ersten Eingabeadresse ist, und
betreibbar, um die zwei Übersetzungen in den Übersetzungspuffer einzusetzen,
welcher Tabellenläufer betreibbar ist, um die Übersetzung, die die zweite Eingabeadresse enthält, die von der Tabelle wieder erlangt wurde, in ein Element des Übersetzungspuffers einzusetzen in Abhängigkeit vom Ausgang einer weiteren Durchsuchung durch den Übersetzungspuffer-Durchsucher nach einer gültigen Übersetzung der zweiten Eingabeadresse in einer solchen Weise, dass
wenn der Übersetzungspuffer-Durchsucher genau eine Übersetzung, die die zweite Eingabeadresse enthält, in einem Element des Übersetzungspuffers findet, die Übersetzung, die die zweite Eingabeadresse enthält, die von der Tabelle wieder erlangt wurde, in jenes Element des Übersetzungspuffers eingesetzt wird, das von dem Übersetzungspuffer-Durchsucher gefunden wurde, und
wenn der Übersetzungspuffer-Durchsucher scheitert, eine gültige Übersetzung, die die zweite Eingabeadresse enthält, in dem Übersetzungspuffer zu finden, die Übersetzung, die die zweite Eingabeadresse enthält, die von der Tabelle wieder erlangt wurde, in jenes Element des Übersetzungspuffers eingesetzt wird, das von der Zeigerlogik (**138**) angegeben wird.

10. Gerät nach Anspruch 9, wobei jede Eingabeadresse ein Kennungsfeld enthält und jede Übersetzung, die in dem Übersetzungspuffer gespeichert ist, auch ein Kennungsfeld enthält, welcher Übersetzungspuffer-Durchsucher so arbeitet, dass eine gespeicherte Übersetzung als eine Übersetzung einer Eingabeadresse nur angesehen wird, wenn die Kennungsfelder der Eingabeadresse und der Übersetzung denselben Wert haben.

11. Gerät nach Anspruch 9 oder 10, wobei das Durchsuchen, das von dem Übersetzungspuffer-Durchsucher nach einer Übersetzung ausgeführt wird, die die zweite Eingabeadresse enthält, eine Zeit, die für den Übersetzungspuffer-Einsetzer zum Einsetzen der Übersetzung, die die zweite Eingabeadresse enthält, in den Übersetzungspuffer erforderlich ist, nicht verlängert.

12. Gerät nach einem der Ansprüche 9 bis 11, wobei die Übersetzungstabelle nicht in einer Seitentabelle gespeichert ist.

13. Gerät nach einem der Ansprüche 9 bis 12, wobei die Tabelle eine B-Baum-Datenstruktur ist, die wenigstens einen Knoten enthält.

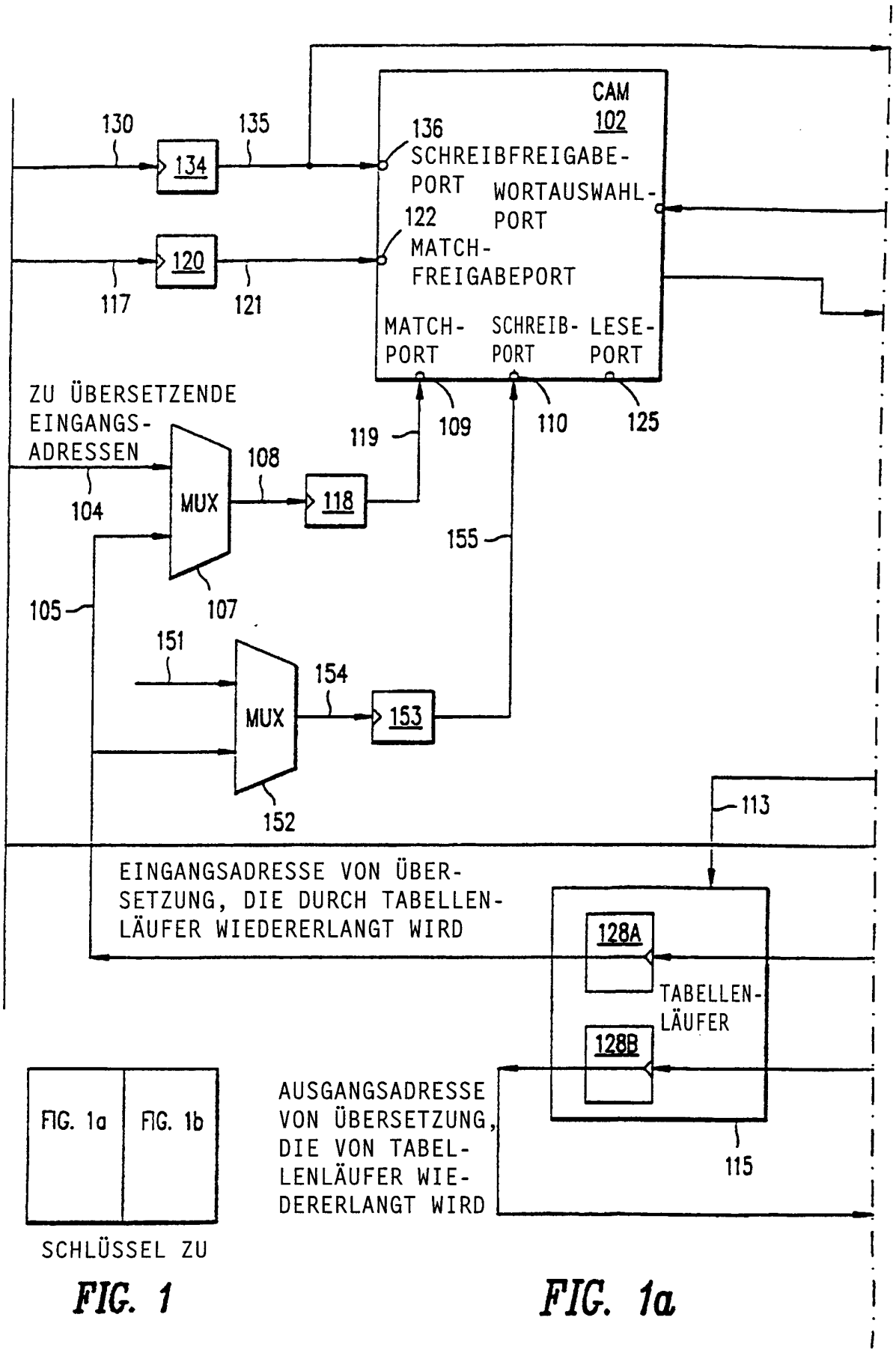
14. Gerät nach Anspruch 13, wobei die B-Baum-Datenstruktur Blattknoten und Indexknoten enthält, jeder der Blattknoten wenigstens eine Übersetzung speichert und jeder der Indexknoten wenigstens einen Zeiger speichert, der zu einem anderen Knoten der B-Baum-Datenstruktur zeigt, welches Gerät ferner enthält:
einen Knotenzeigeruntersucher, wodurch der Knotenzeigeruntersucher bestimmt, ob ein Knoten der B-Baum-Datenstruktur, auf den durch das Knotenadressenfeld eines Knotenzeigers gezeigt wird, ein Blattknoten oder ein Indexknoten ist;
einen Indexknoten-Wiedererlanger/-Prozessor, der operativ an den Speicher gekoppelt ist, der die Tabelle und den Knotenzeigeruntersucher enthält, wodurch der Indexknoten-Wiedererlanger/-Prozessor von dem Speicher wieder erlangt und verarbeitet den Knoten der B-Baum-Datenstruktur, wenn der Knotenzeigeruntersucher bestimmt, dass der Knoten der B-Baum-Datenstruktur ein Indexknoten ist; und
einen Blattknoten-Wiedererlanger/-Prozessor, der operativ an den Speicher gekoppelt ist, der die Tabelle und den Knotenzeigeruntersucher enthält, wodurch der Blattknoten-Wiedererlanger/-Prozessor von dem Speicher wieder erlangt und verarbeitet den Knoten der B-Baum-Datenstruktur, wenn der Knotenzeigeruntersucher bestimmt, dass der Knoten der B-Baum-Datenstruktur ein Blattknoten ist;
wobei der Blattknoten-Wiedererlanger/-Prozessor den Knoten wieder erlangt und in dem Fall des Einsetzens einer Übersetzung davon in den Übersetzungspuffer jede Übersetzung, die in dem betroffenen Blattknoten gespeichert ist, in den Übersetzungspuffer eingesetzt wird.

15. Gerät nach Anspruch 13 oder 14, wobei die Übersetzung, die die spezifizierte erste Eingabeadresse enthält, und die Übersetzung, die die zweite Eingabeadresse enthält, in einem selben Knoten der B-Baum-Datenstruktur gespeichert sind.

16. Gerät nach Anspruch 13, 14 oder 15, wobei eine Größe von wenigstens einem Knoten der B-Baum-Datenstruktur gleich einer Cache-Leitungsgröße des Speichers ist.

Es folgen 11 Blatt Zeichnungen

Anhängende Zeichnungen



SCHLÜSSEL ZU

FIG. 1

FIG. 1a

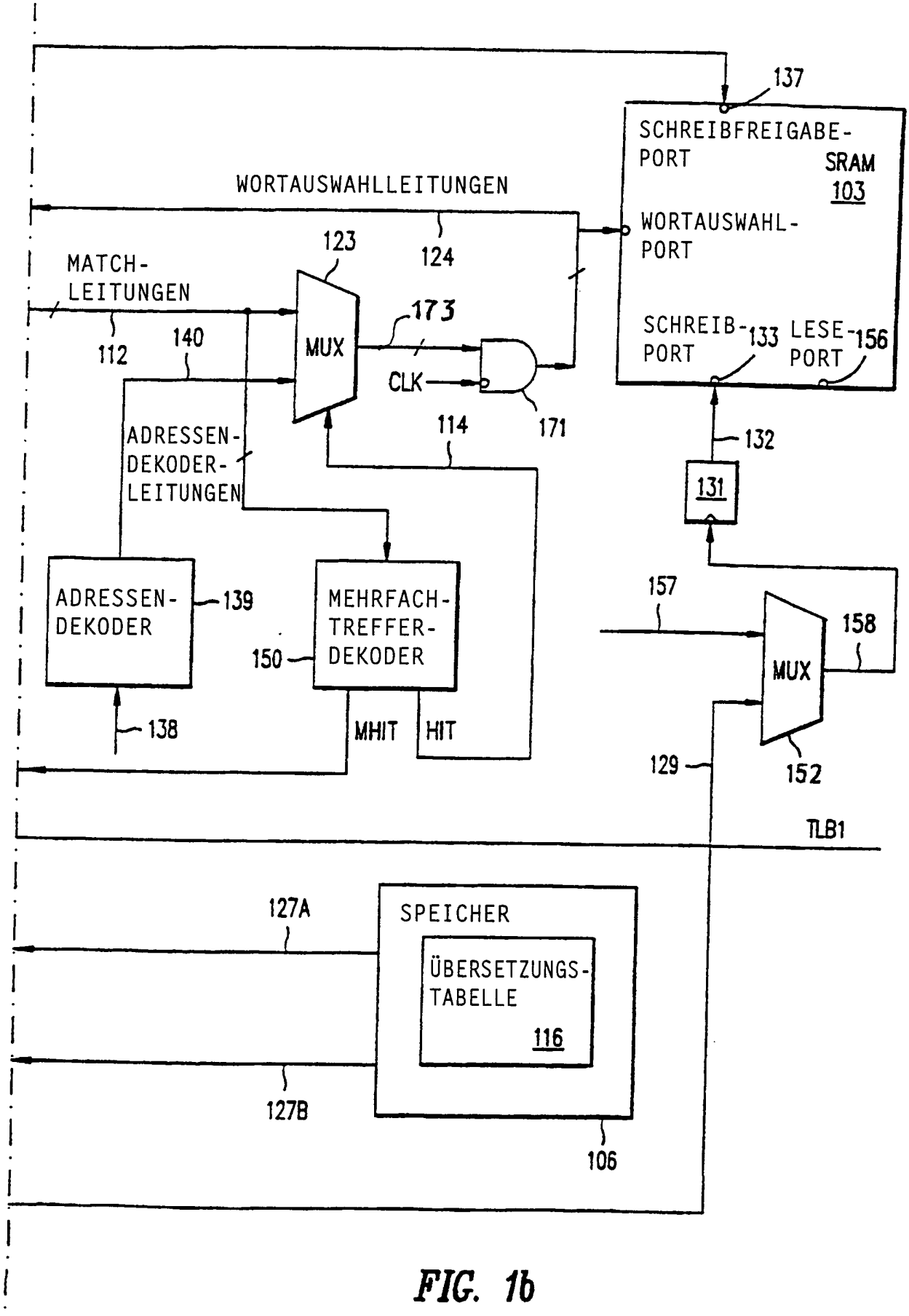

137
SCHREIBFREIGABE-PORT
SRAM 103
WORTAUSWAHLLEITUNGEN
124
WORTAUSWAHL-PORT
123
MATCH-LEITUNGEN
173
112
140
MUX
SCHREIB-PORT
133
LESE-PORT
156
CLK
171
114
132
ADRESSEN-DEKODER-LEITUNGEN
131
ADRESSEN-DEKODER
139
MEHRFACH-TREFFER-DEKODER
150
157
158
MUX
138
MHIT
HIT
129
152
TLB1
127A
SPEICHER
ÜBERSETZUNGS-TABELLE
116
127B
106

FIG. 1b

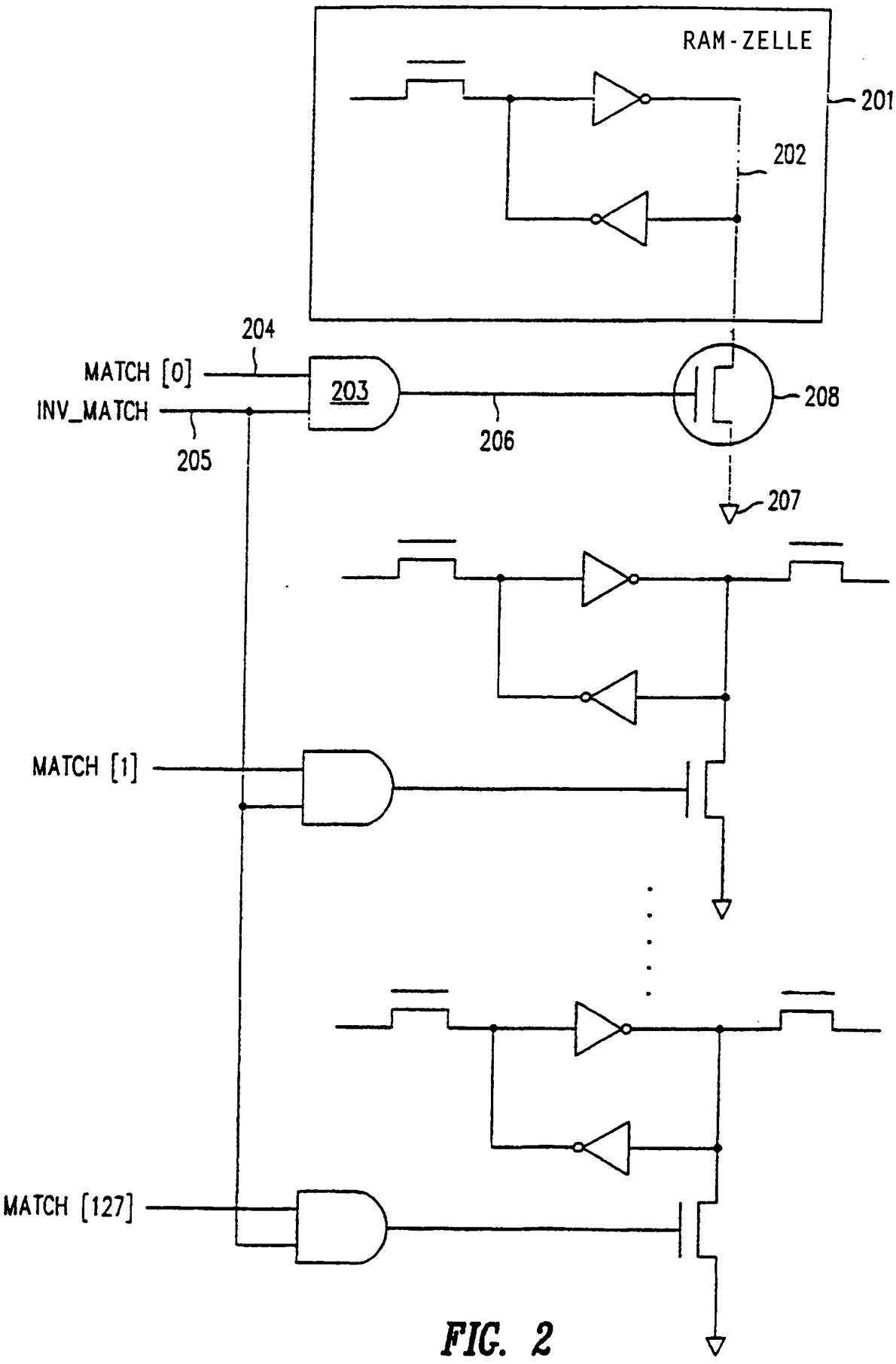

RAM-ZELLE
201
202
204
MATCH [0]
INV_MATCH
205
203
206
208
207
MATCH [1]
MATCH [127]

FIG. 2

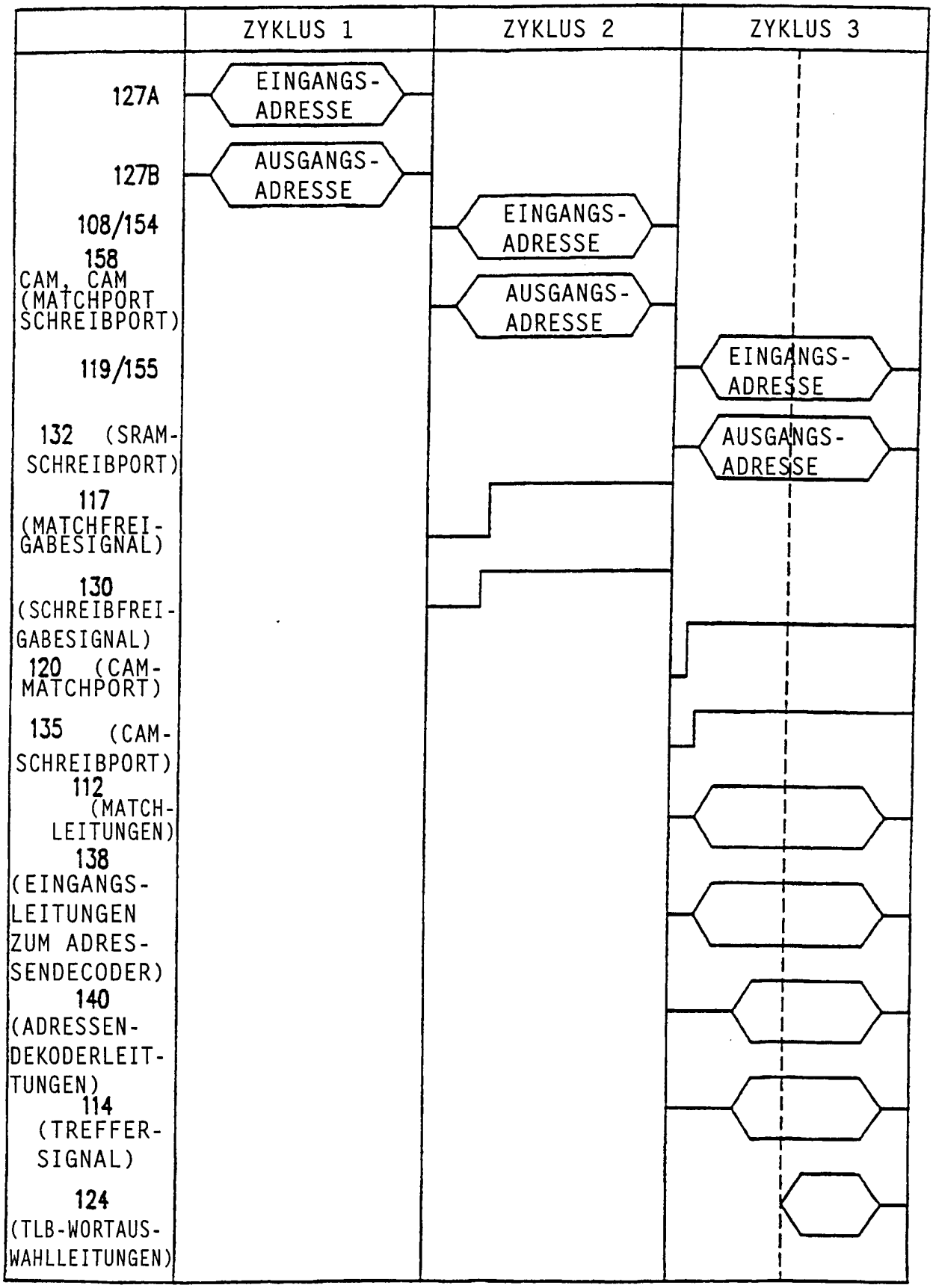

ZYKLUS 1
ZYKLUS 2
ZYKLUS 3
127A
EINGANGS-ADRESSE
127B
AUSGANGS-ADRESSE
108/154
158
CAM, CAM (MATCHPORT SCHREIBPORT)
EINGANGS-ADRESSE
AUSGANGS-ADRESSE
119/155
EINGANGS-ADRESSE
132 (SRAM-SCHREIBPORT)
AUSGANGS-ADRESSE
117 (MATCHFREI-GABESIGNAL)
130 (SCHREIBFREI-GABESIGNAL)
120 (CAM-MATCHPORT)
135 (CAM-SCHREIBPORT)
112 (MATCH-LEITUNGEN)
138 (EINGANGS-LEITUNGEN ZUM ADRES-SENDECODER)
140 (ADRESSEN-DEKODERLEIT-TUNGEN)
114 (TREFFER-SIGNAL)
124 (TLB-WORTAUS-WAHLLEITUNGEN)

FIG. 3

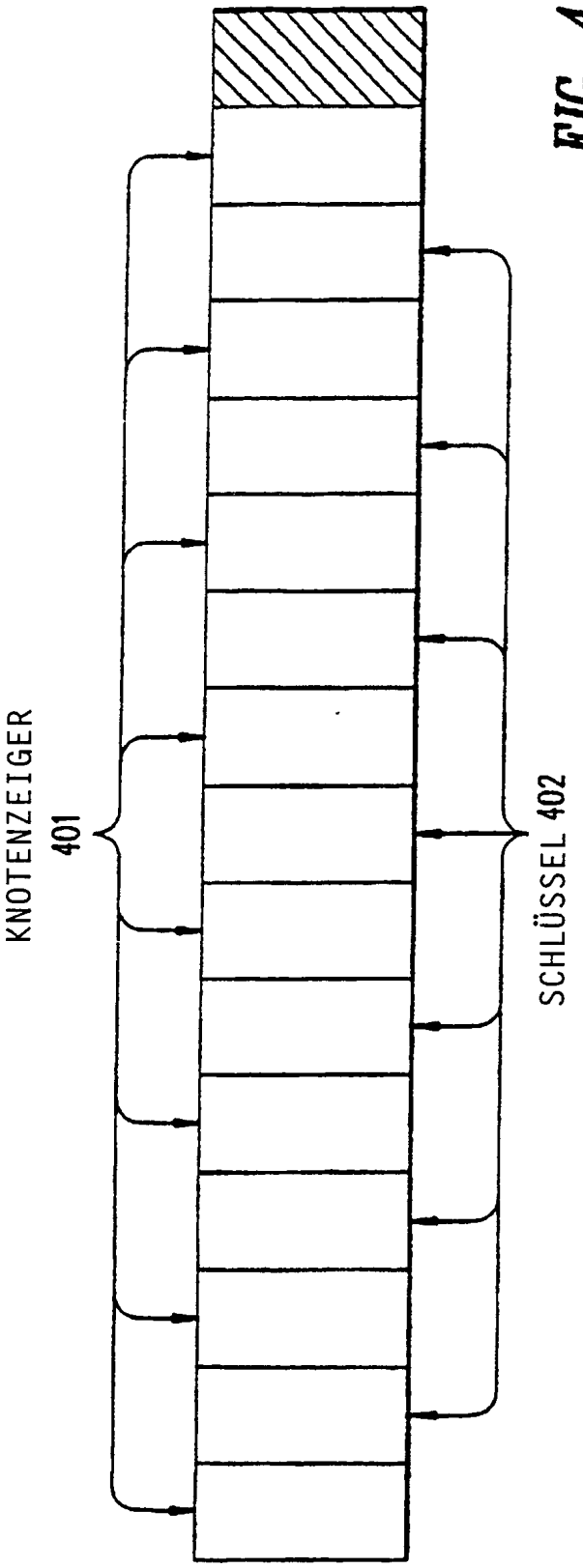

KNOTENZEIGER
401
SCHLÜSSEL 402

FIG. 4

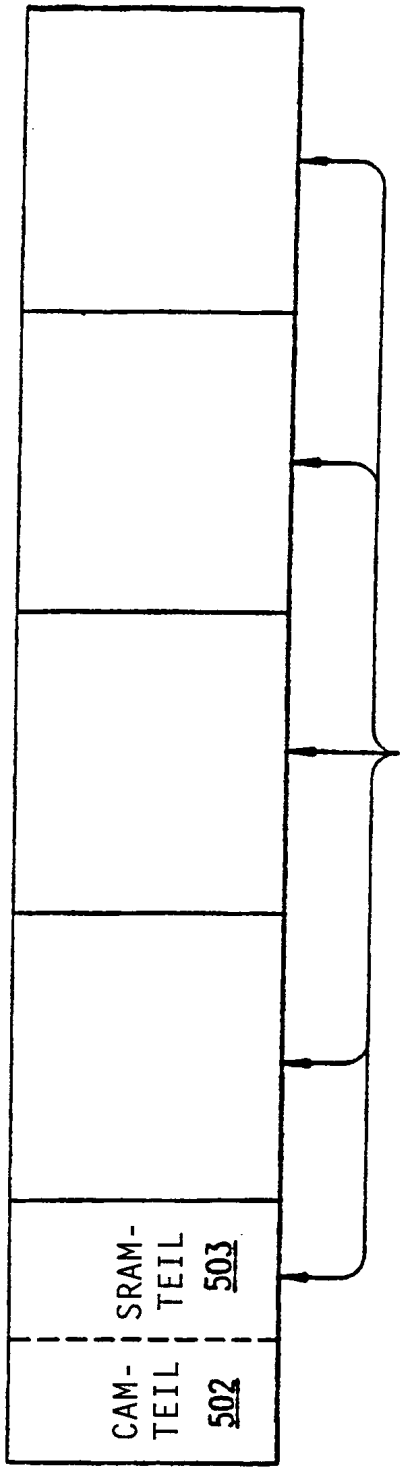

CAM-TEIL 502
SRAM-TEIL 503
ÜBERSETZUNGEN
501

FIG. 5

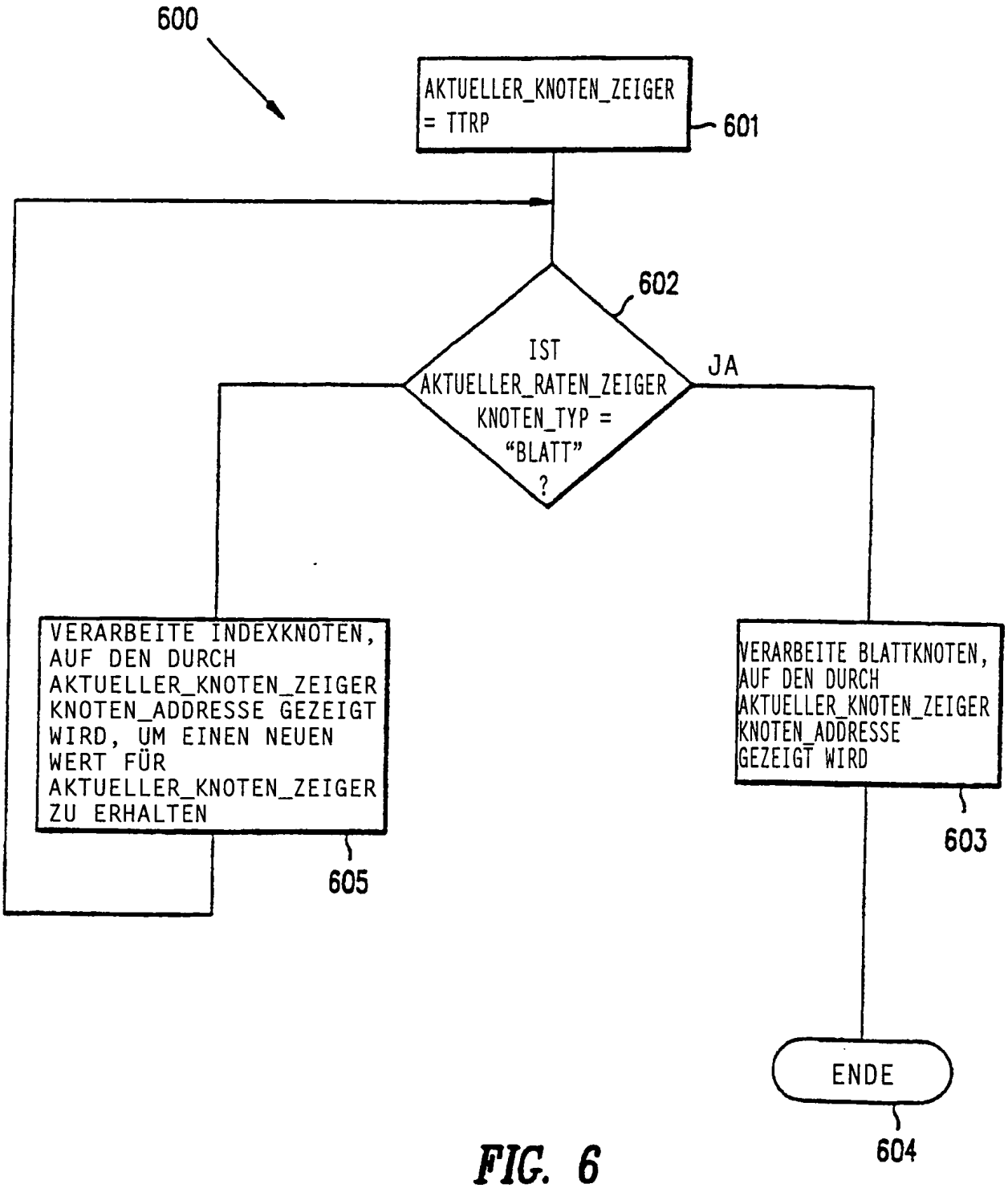

600
AKTUELLER_KNOTEN_ZEIGER = TTRP
601
602
IST AKTUELLER_RATEN_ZEIGER KNOTEN_TYP = "BLATT" ?
JA
VERARBEITE INDEXKNOTEN, AUF DEN DURCH AKTUELLER_KNOTEN_ZEIGER KNOTEN_ADDRESSE GEZEIGT WIRD, UM EINEN NEUEN WERT FÜR AKTUELLER_KNOTEN_ZEIGER ZU ERHALTEN
605
VERARBEITE BLATTKNOTEN, AUF DEN DURCH AKTUELLER_KNOTEN_ZEIGER KNOTEN_ADDRESSE GEZEIGT WIRD
603
ENDE
604

FIG. 6

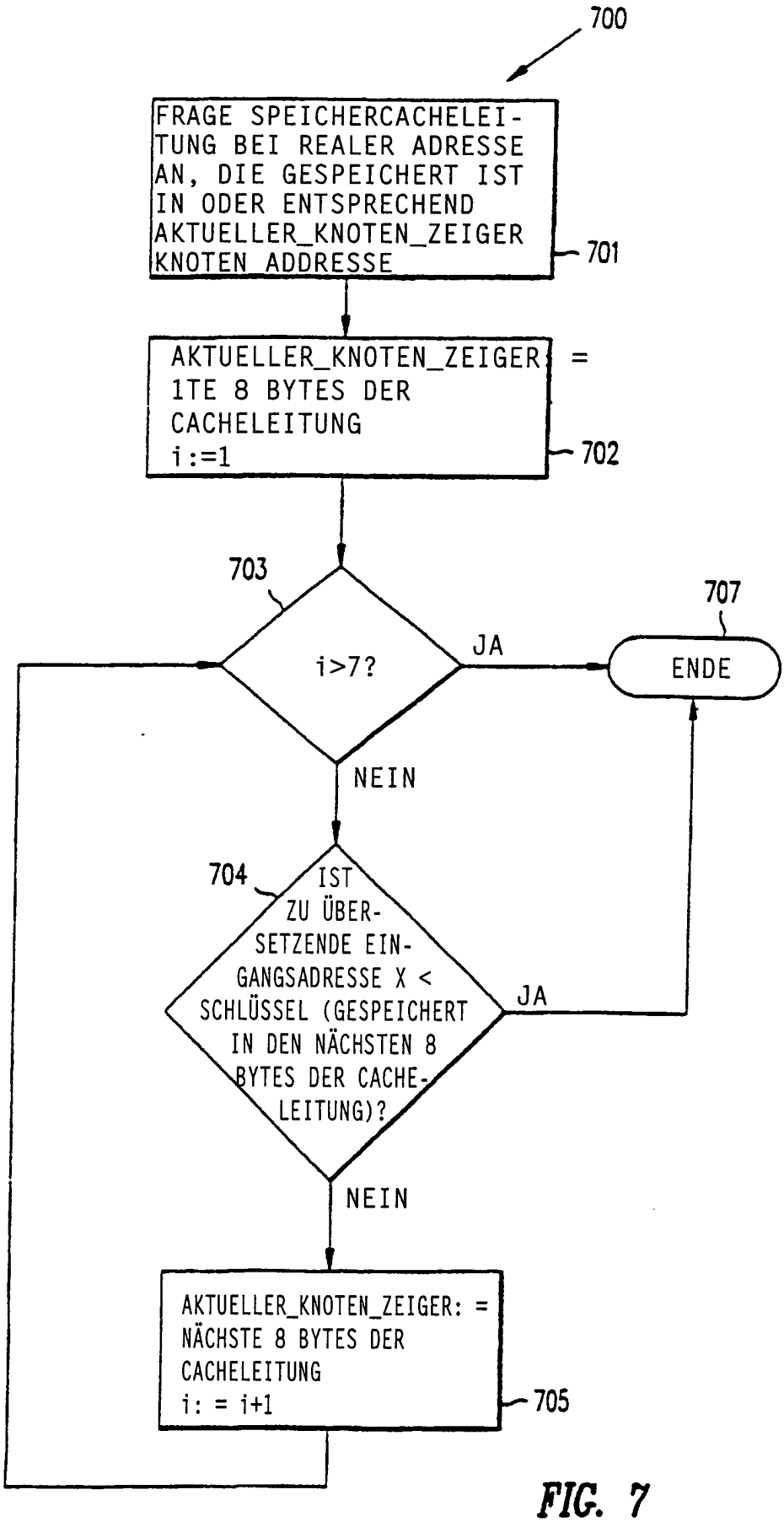

700
FRAGE SPEICHERCACHELEITUNG BEI REALER ADRESSE AN, DIE GESPEICHERT IST IN ODER ENTSPRECHEND AKTUELLER_KNOTEN_ZEIGER KNOTEN ADDRESSE
701
AKTUELLER_KNOTEN_ZEIGER = 1TE 8 BYTES DER CACHELEITUNG i:=1
702
703
i>7?
JA
707
ENDE
NEIN
704
IST ZU ÜBERSETZENDE EINGANGSADRESSE X < SCHLÜSSEL (GESPEICHERT IN DEN NÄCHSTEN 8 BYTES DER CACHELEITUNG)?
JA
NEIN
AKTUELLER_KNOTEN_ZEIGER: = NÄCHSTE 8 BYTES DER CACHELEITUNG i: = i+1
705

FIG. 7

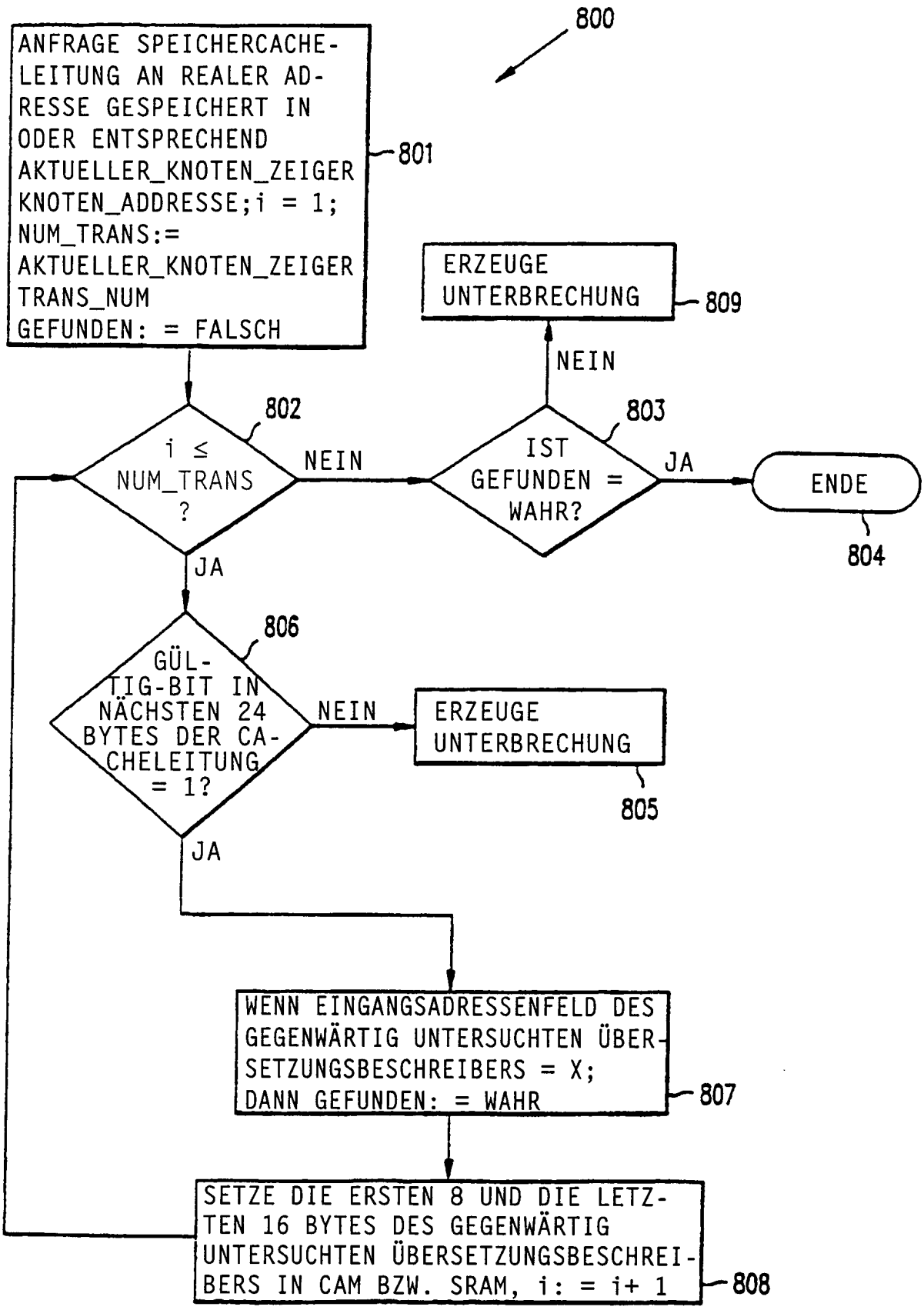

800
ANFRAGE SPEICHERCACHE-LEITUNG AN REALER ADRESSE GESPEICHERT IN ODER ENTSPRECHEND AKTUELLER_KNOTEN_ZEIGER KNOTEN_ADDRESSE;i = 1; NUM_TRANS:= AKTUELLER_KNOTEN_ZEIGER TRANS_NUM GEFUNDEN: = FALSCH
801
ERZEUGE UNTERBRECHUNG
809
NEIN
802
i ≤ NUM_TRANS ?
NEIN
803
IST GEFUNDEN = WAHR?
JA
ENDE
804
JA
806
GÜLTIG-BIT IN NÄCHSTEN 24 BYTES DER CACHELEITUNG = 1?
NEIN
ERZEUGE UNTERBRECHUNG
805
JA
WENN EINGANGSADRESSENFELD DES GEGENWÄRTIG UNTERSUCHTEN ÜBERSETZUNGSBESCHREIBERS = X; DANN GEFUNDEN: = WAHR
807
SETZE DIE ERSTEN 8 UND DIE LETZTEN 16 BYTES DES GEGENWÄRTIG UNTERSUCHTEN ÜBERSETZUNGSBESCHREIBERS IN CAM BZW. SRAM, i: = i+ 1
808

FIG. 8

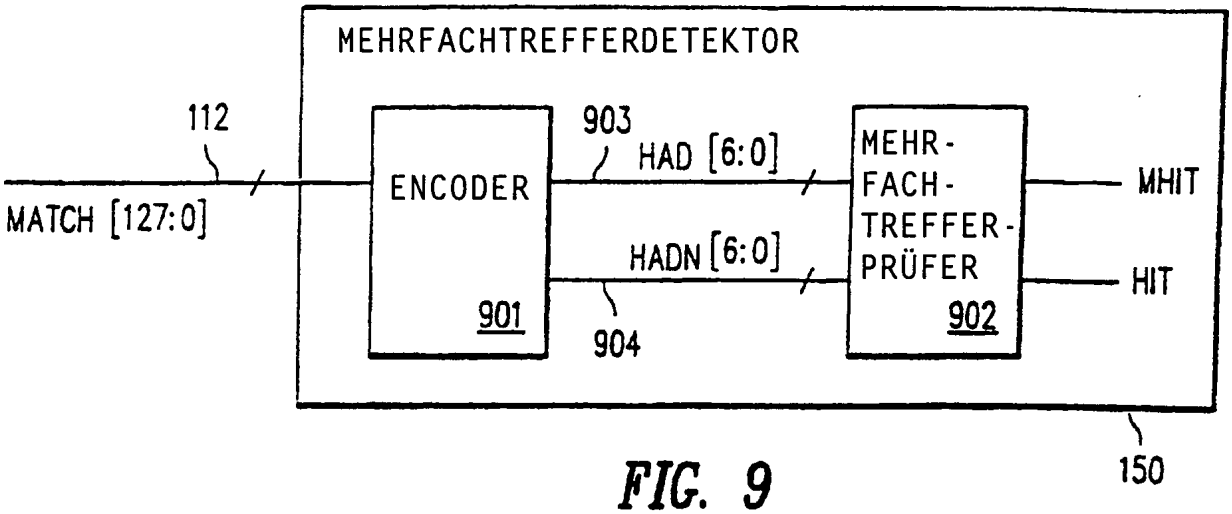

MEHRFACHTREFFERDETEKTOR
112
MATCH [127:0]
ENCODER
901
903
HAD [6:0]
HADN [6:0]
904
MEHR-
FACH-
TREFFER-
PRÜFER
902
MHIT
HIT
150

FIG. 9

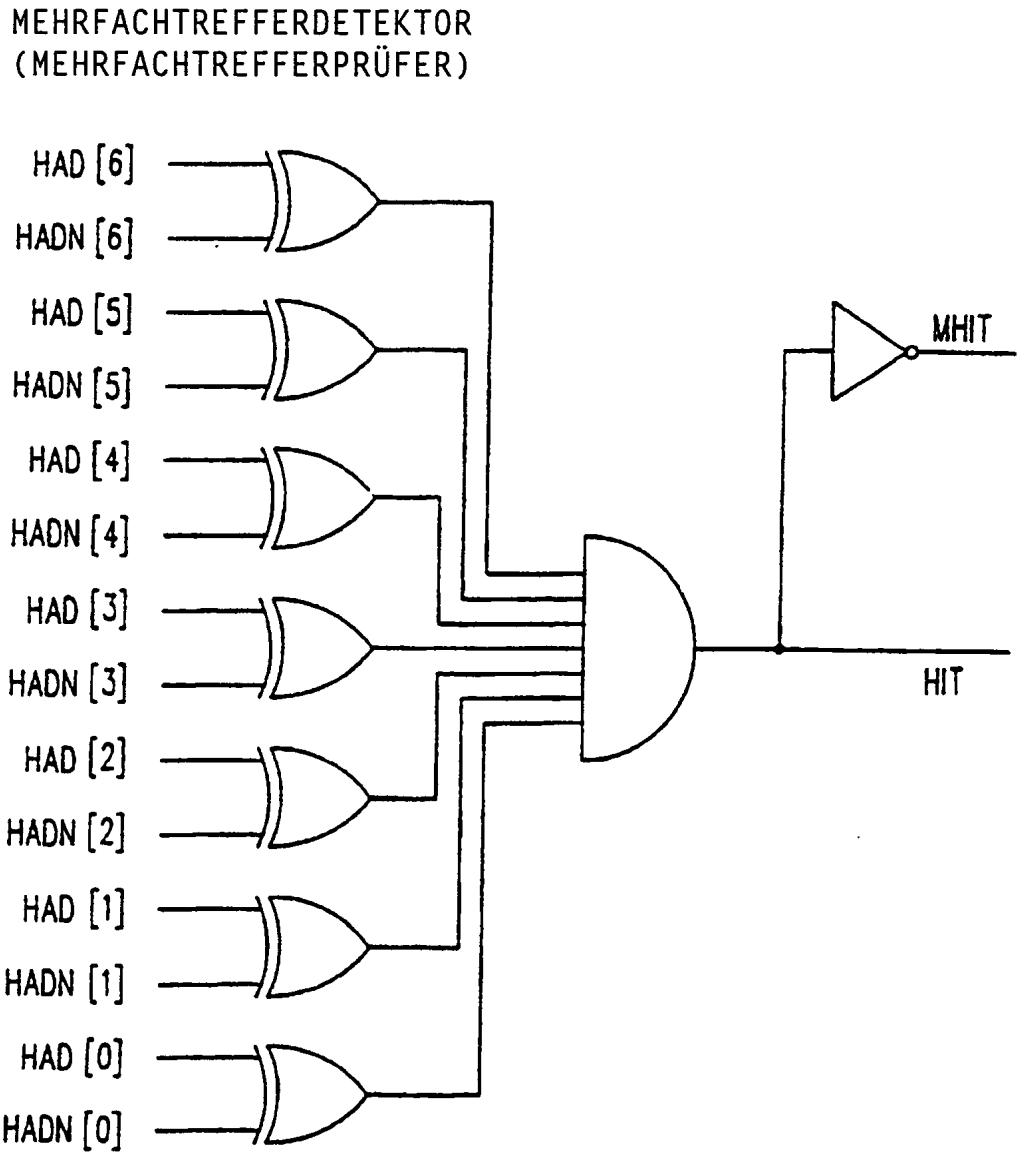

MEHRFACHTREFFERDETEKTOR
(MEHRFACHTREFFERPRÜFER)
HAD [6]
HADN [6]
HAD [5]
HADN [5]
HAD [4]
HADN [4]
HAD [3]
HADN [3]
HAD [2]
HADN [2]
HAD [1]
HADN [1]
HAD [0]
HADN [0]
MHIT
HIT

FIG. 11

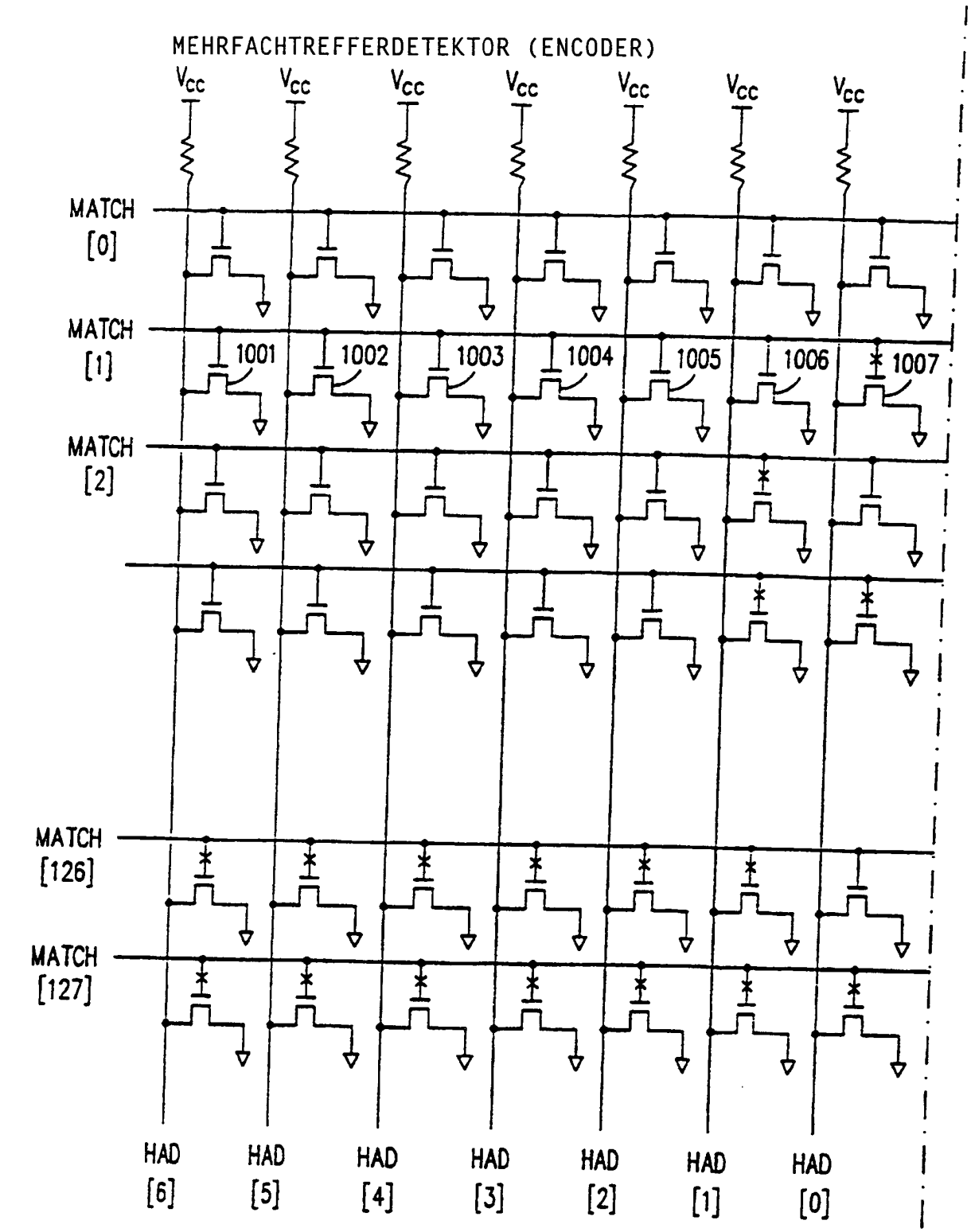


FIG. 10a

SCHLÜSSEL ZU

FIG. 10a	FIG. 10b

FIG. 10

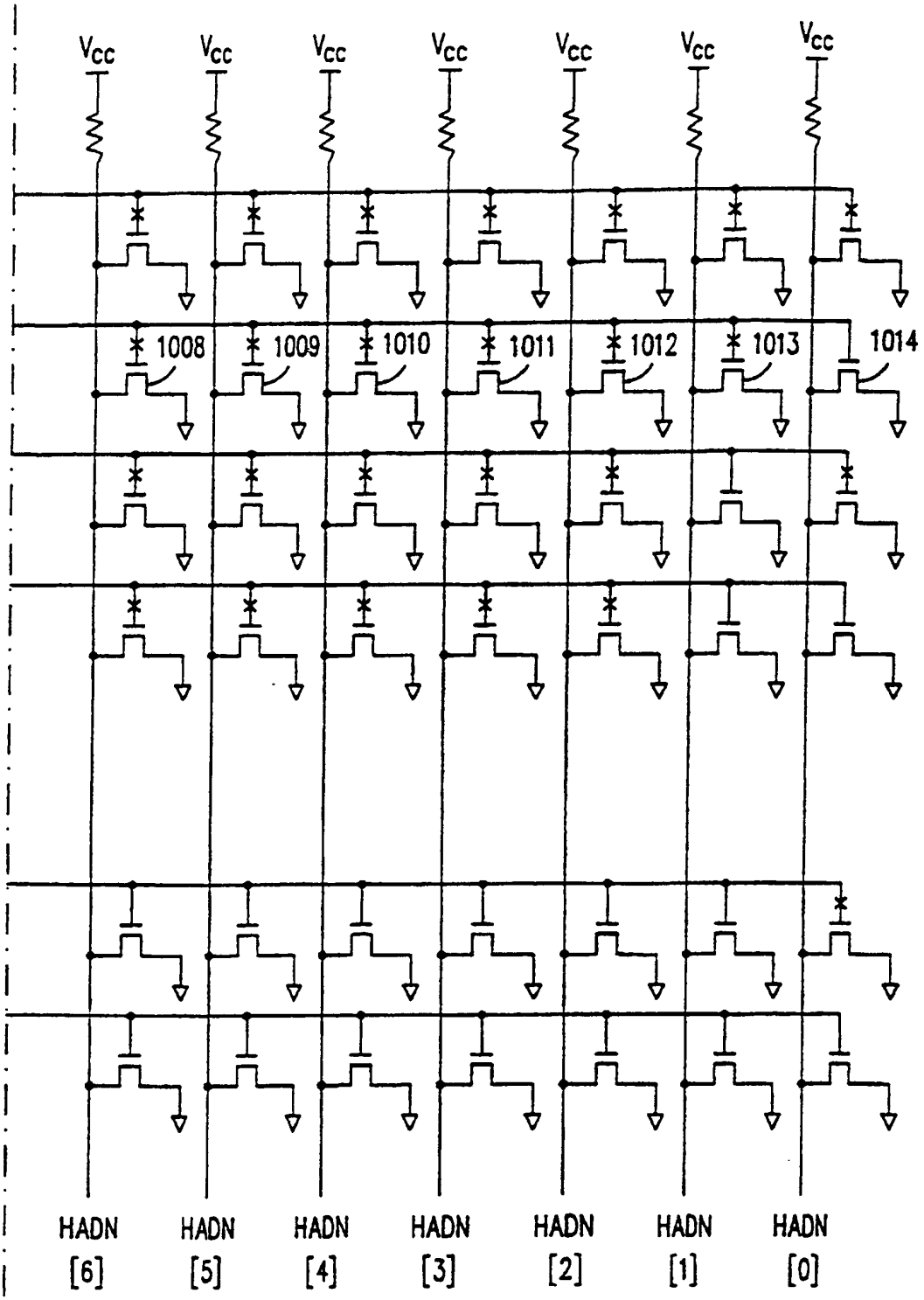

Vcc
Vcc
Vcc
Vcc
Vcc
Vcc
Vcc
1008
1009
1010
1011
1012
1013
1014
HADN [6]
HADN [5]
HADN [4]
HADN [3]
HADN [2]
HADN [1]
HADN [0]

FIG. 10b